

GE Fanuc Automation

可编程控制器产品

系列 90™-30/20/Micro PLC
CPU 指令集

参考手册

GFK-0467M
May 2002

GFL-002

在本出版物中使用的
警告、当心和注意标志

警告

在该出版物中，警告标志被用来强调危险电压、电流、温度或其他条件，这些危险条件可能造成设备或与其使用有关的人身伤害。

在不加注意就会造成人身伤害或设备损坏的情况下，使用“警告”标志。

当心

在因疏忽会可能损坏设备的位置，使用“当心”标志。

注意

“注意”只是引起对理解和操作设备特别重要的信息的注意。

本手册的内容基于该版本出版时可以得到的信息。尽管内容力求精确，但是难以涵盖软硬件所有细节和变更信息，也不能提供在安装、运行或维护中遇到所有可能的意外情况。本书没有描述所有的硬件和软件系统的全部性能。如今后有关信息发生变化，GE Fanuc Automation 没有义务通知本书的持有者。

GE Fanuc Automation 公司没有任何表示或保证，明确或暗示，也就是，就法律规定而言，对本资料中所包含信息的准确性、完整性和实用性，本公司不承担责任。不保证它的可做商品或适合应用的目的。

下面是 GE Fanuc Automation 北美公司的一些注册商标.

Alarm Master
Genius
PROMACRO
Series Six CIMPLICITY
Helpmate
PowerMotion
Series Three
CIMPLICITY 90–ADS
Logicmaster
PowerTRAC
VersaMax CIMSTAR
Modelmaster
Series 90
VersaPro Field Control
Motion Mate
Series Five
VuMaster GEnet
ProLoop
Series One
Workmaster

前言

本手册主要介绍系列 90™-30, 系列 90-20 和系列90 Micro PLC系统运行，故障处理，和 Logicmaster 90™ 编程指令。系列90-30 PLCs, 系列 90-20 PLCs, 和系列 90 Micro PLCs 是 GE Fanuc Automation PLC 90系列家族成员。

手册修订说明

- 增强型374 CPU, 支持以太网连接，有两个内置10BaseT/100BaseTx 自适应全双工以太网接口。 型号 364 (版本9.10 及以后) 和 374 是系列 90-30 CPUs 中唯一支持EGD的CPU。注意CPU374仅支持基于Windows®编程器。
- 其 他 需 要 的 修 正 和 说 明 。

相关手册

Logicmaster™ 90 系列 90™-30/20/Micro 编程软件用户手册 (GFK-0466).

VersaPro™编程软件用户手册 (GFK-1670)

CIMPLICITY® Machine Edition Getting Started (GFK-1868)

系列 90™-30 控制器安装手册 (GFK-0356)

系列90™-20控制器安装手册 (GFK-0551)

系列90™-30 I/O 模块使用手册 (GFK-0898)

系列90™ 可编程协处理器模块和软件用户手册(GFK-0255)

系列 90™ PCM 开发软件 (PCOP) 用户手册 (GFK-0487)

CIMPLICITY™ 90-ADS 文字数字显示系统用户手册 (GFK-0499)

CIMPLICITY™ 90-ADS文字数字显示系统参考手册 (GFK-0641)

系列 90™-30 and 90-20 PLC 手持编程器用户手册 (GFK-0402)

Power Mate APM系列 90™-30 PLC— 标准模式用户手册 (GFK-0840)

Power Mate APM系列 90™-30 PLC—Follower Mode用户手册 (GFK-0781)

Motion Mate™ DSM302系列90™-30 PLCs用户手册 (GFK-1464)

系列 90™-30 高速计数器用户手册(GFK-0293)

系列 90™-30 Genius 通讯模块用户手册 (GFK-0412)

前言

系列 90™-30 Genius™ 总线控制器用户手册 (GFK-1034)

系列 90™-70 FIP总线控制器用户手册(GFK-1038)

系列 90™-30 FIP 远程 I/O扫描用户手册 (GFK-1037)

Field Control™ 分布式 I/O和控制系统 Genius™ 总线接口单元用户手册(GFK-0825)

系列 90™ Micro PLC用户书册 (GFK-1065)

系列 90™ PLC串口通讯用户手册 (GFK-0582)

目录

目录

目录

Chapter 1

第一章　绪言

系列90-30, 90-20, 和Micro PLCs 是GE Fanuc系列90可编程控制器(PLCs)家族成员。这系列PLC易于安装和配置，具有先进的编程性能，且与系列90-70 PLCs兼容。

341和系列90-30较低档PLCs和系列90-20 PLC使用80188微处理器。系列90-30 PLCs中35x和36x使用80386EX微处理器。系列90-30 PLCs中37x使用586微处理器。系列90 Micro PLC使用H8 微处理器。支持程序执行和内务管理诸如诊断程序，输入/输出扫描，报警处理等。系统还支持和编程器通讯。和编程器通讯包括上传和下载应用程序，返回状态信息，和控制PLC等。

系列90-30 PLC, 应用程序(用户逻辑)控制工作过程，而其又是由专门的指令协处理器(ISCP)控制。ISCP由313和更高级硬件和311和Micro PLC软件实现。微处理器和基于硬件的ISCP能同时执行，当ISCP执行应用程序同时，允许微处理器提供通信，然而微处理器必须执行非布尔指令功能。

当出现的故障或状况影响系统运行和执行时，表示系列90-30 PLC, 系列90-20 PLC, 和Micro PLC故障。一些故障可能会影响PLC控制机器或过程。其他状况只是作为一种报警，例如电池电压低信号指示应更换电池。这些状况或故障统称为故障。

这些故障由软件报警处理器处理，它将这些故障记录到PLC或I/O故障表中去(331型和更高级CPUs按时间顺序记录这些故障)。这些故障表通过编程软件中的PLC故障表和I/O故障表显示出来，在Logicmaster 90-30/20/Micro 软件中使用控制和状态功能。

注意

只有35x和36x系列CPUs 支持浮点运算，且CPU352和CPU374所有版本都支持，其他CPUs版本9及以后版本才支持。

CPU364 (版本9.10或以后) 和CPU374 是系列90-30 CPUs 中支持全局数据交换(EGD)的两种CPU。

系列90-20 PLC 为I/O点数较少的应用提供高效低成本的平台。系列90-20 PLC 基本特征如下：

- 提供易于使用，安装，升级和维护的小型PLC
- 提供高效低成本可兼容的家族PLC
- 通过标准的通讯硬件和习武一提供更简单的系统集合。

系列90 Micro PLC也为I/O点数较少的应用提供高效低成本的平台。Micro PLC基本特征与系列90-20相同。另外，还包括以下特征：

- Micro PLC是CPU, 电源, 输入输出一体的小型设备
- 多数型号有一高速计数器
- 因为CPU, 电源, 和输入输出都集成到一个设备，所以易于配置。

注意

附加参考信息，参见本手册后面的附录.

- 附录 A 列出编程指令所占内存字节数和执行时间，时间以微秒为单位。
- 附录 B 描述PLC和I/O故障表故障信息结构形式。
- 附录 C 列出在程序中寻找或编辑时用到的指令助记符。
- 附录 D 列出Logicmaster 90-30/20/Micro软件特殊键盘说明。
- 附录 E 描述浮点运算的使用。

致基于Windows PLC编程软件用户

本手册是为使用Logicmaster (基于DOS PLC编程软件)用户编写的。基于Windows PLC软件产品,例如CIMPLICITY® Machine Edition 逻辑开发软件和VersaPro®, 提供PLC指令在线帮助且胜于手册。基于Windows编程软件用户应该知道指令和Logicmaster 中看起来不同('它们在PLC中作用一样)。基于Windows的编程软件的在线帮助关于指令提供最准确的信息。关于这两种编程软件主要区别，参见附录F。

第二章 *系统操作*

本章描述系列90-30, 90-20, 和Micro PLC系统特定的系统操作。这些系统操作包括：

第一节: PLC扫描概述 ：

系列90-30, 90-20和Micro PLC中，程序都是以循环的方式执行直到被编程器停止或者被其他设备停止。每次执行一个程序所需的操作时序称之为一次扫描。除了执行逻辑程序，扫描还包括从输入设备中获得数据，发送数据到输出设备，执行内部系统管理，为编程器服务和为其它通讯服务。

系列90-30, 90-20和Micro PLC通常以标准程序扫描模式运行。其它操作模式包括：I/O中断模式下停止，I/O使能方式下停止和恒定扫描模式。本章所描述的每种扫描方式都是由外部过程和应用配置的设定来控制的。在每次扫描开始时，PLC都要确定这次操作模式的决定。

标准程序扫描：

标准程序扫描方式通常可在所有的状态下运行。CPU执行应用程序，更新I/O，执行通讯和其他任务。以上过程循环执行叫做CPU扫描。标准程序的执行顺序共分七个部分：

1. 系统管理的起始扫描
2. 输入扫描（读取输入）
3. 执行应用程序
4. 输出扫描（更新输出）
5. 和编程器的通讯
6. 系统通讯
7. 诊断

以上步骤每次都执行。尽管编程器通讯服务每个扫描周期都打开，但是只有当检测到线路板故障或编程设备发出服务请求时才执行；也就是，编程通讯服务首先检查是否有服务请求或故障。下图是标准程序的扫描顺序：

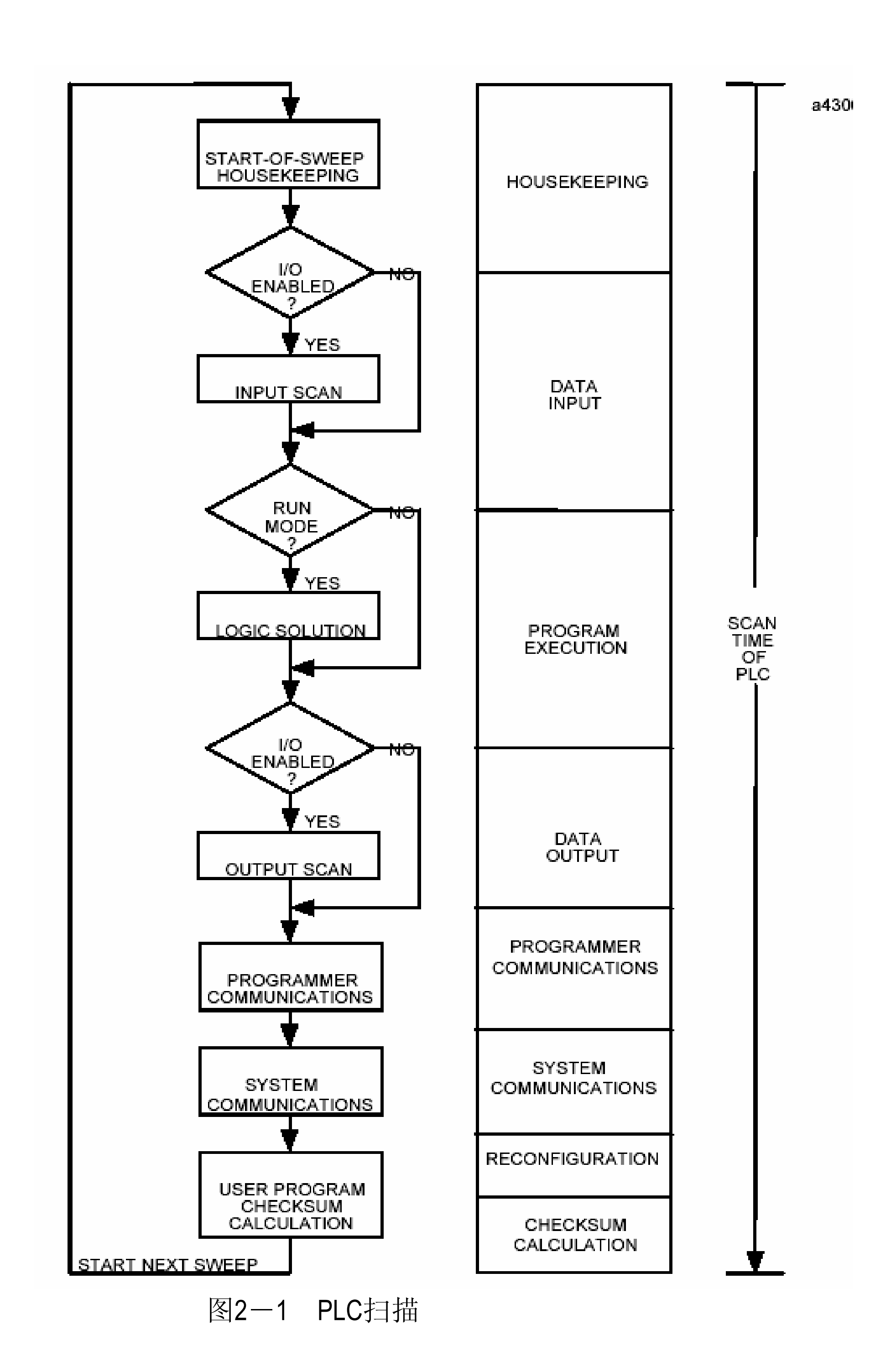


图2－1　PLC扫描

正如PLC扫描步骤所示，扫描中包括多项。这些项目在整个扫描中所占的时间如下：

表2－1　扫描时间

扫描步骤	说明		时间分配（毫秒）						
			Micro	211	311/313	331	34x	35x/36x	37x
系统管理	1. 计算扫描时间 2. 计划下一次扫描起点 3. 确定下次扫描模式 4. 更新故障表 5. 复位看门狗定时器		0.368	0.898	0.714	0.705	0.424	0.279	0.027
数据输入	从输入和可选模块中接收数据输入			参照表2-2和2-3（关于扫描时间）					
程序执行	执行用户逻辑		执行时间取决于程序的长度和程序中使用的指令类型，指令执行行时间在附录A中						
数据输出	发送输出数据到输出模块或可选模块		0．1656	参看表2－2 中扫描时间分配					
编程器和系统通讯	处理来自编程器设备和智能模块服务请求	HHP	1.93	6.526	4.426	4.524	2.476	0.334	N/A
		编程器	0.380	3.536	2.383	2.454	1.248	0.517	0.026
	PCM	PCM2	N/A	N/A	N/A	3.337	1.974	0.482	0.029
重新组态	监视带有故障模块的插槽和空插槽		N/A6	N/A	0.458	0.639	0. 463	0.319	0.243
诊断	核对用户程序的完整性		N/A	0.083	0.050	0.048	0.031	0.010	0.022

N/A

1. 对外部设备的服务对扫描时间的影响依赖于服务进程中通讯窗口模式，假如窗口模式为限制模式，对311，313，323和331CPU来说，最多耗时为8毫秒，对340和更高版本的CPU来说最多耗时为6毫秒。在窗口模式为完全运行模式时，窗口中能够耗用的时间最多为50毫秒，这取决于现有的请求服务数量。
2. 那些测量被应用在PCM，但没有被配置和应用任务在PCM中。
3. 用SVCREQ函数模块能够更改每次扫描校验和字数。
4. 那些度量被应用在一个空的程序中和默认的配置中，在90-30系列PLC有10槽机架没有扩展机架。
5. Micro PLC中数据输入时间由以下决定：0.365ms（混合扫描）+0.036ms（过滤时间）*（总的扫描时间）/0.5ms.
6. 一旦PLC中有了一个静态的I/O设置，不需重新设置。
7. 假如用户的Micro PLC程序在闪存中，它将不会检查它的完整性。

表 2-2. 系列90-30 35x, 36x 和 37x CPUs I/O 扫描时间（毫秒）

模块类型		35x and 36x 系列 CPUs			37x系列 CPUs		
		主机架	扩展机架	远程机架	主机架	扩展机架	远程机架
8位离散输入		.030	.055	.206	.030	.055	.206
16位离散输入		.030	.055	.206	.030	.055	.206
32位离散输入		.043	.073	.269	.048	.075	.272
8位离散输出		.030	.053	.197	.024	.052	.198
16位离散输出		.030	.053	.197	.030	.052	.199
32位离散输出		.042	.070	.259	.047	.069	.258
混合离散输入输出		.060	.112	.405	.052	.110	.408
4通道模拟量输入		.075	.105	.396	.085	.109	.403
2通道模拟量输出		.058	.114	.402	.046	.101	.393
16通道模拟量输入 (电流或电压)		.978	1.446	3.999	.423	.700	1.741
8通道模拟量输出		1.274	1.988	4.472	.873	1.492	3.635
混合模拟量输入输出		1.220	1.999	4.338	.862	1.487	4.103
高速计数器		1.381	2.106	5.221	1.142	1.808	5.234
I/O 处理器		1.574	2.402	6.388	1.270	2.125	6.269
以太网接口 (无连接)		.7129	2.067	3.681	.426	.795	2.302
电源辅助转换管理 (1-轴)		1.527	2.581	6.388	1.236	2.073	6.032
电源辅助转换管理 (2-轴)		1.807	2.864	7.805	1.539	2.439	7.369
DSM 302 *	40 AI, 6 AQ	2.143	3.315	9.527	1.801	2.963	9.275
	50AI, 9 AQ	2.427	3.732	11.092	2.075	3.373	10.840
	64 AI, 12 AQ	2.864	4.317	13.138	2.441	3.931	12.881
DSM314 *	1轴配置	1.6	2.6	6.9	1.330	2.337	6.905
	2轴配置	2.2	3.8	9.9	1.888	3.148	9.917
	3轴配置	2.8	4.3	13.0	2.421	3.953	12.929
	4轴配置	3.3	5.2	15.9	2.969	4.761	15.982
GCM	8 32位设备	8.826	16.932	21.179	7.386	9.520	20.591
GCM+	无设备	.567	.866	1.830	.457	.759	1.743
	32 64字 设备	19.497	25.588	80.871	17.036	24.390	80.044
GBC	无设备	.798	1.202	2.540	.544	.908	2.209
	16 64-字 设备	29.976	40.570	131.702	26.976	38.564	130.639
PCM 311	未配置,或无应用任务	.476	N/A	N/A	.195	N/A	N/A
	运行 20Kb 应用程序	1.746	N/A	N/A	.538	N/A	N/A
ADC (无任务)		.476	N/A	N/A	.193	N/A	N/A
I/O link 主设备	无设备	.569	.865	1.932	.996	1.618	3.749
	16 64-位 设备	4.948	7.003	19.908	5.924	8.240	26.637
I/Olink 从设备	32-位	.087	.146	.553	.095	.149	.540
	64-位 .154	.213	.789	.165	.219	.803	

*对于应用程序，DSM对扫描时间的影响将影响到机器的运行你需要用DO I/O功能块。延迟I／O和快速访问背板总线服务请求发送和获得必要的在每次扫描过程中没有获得所有数据。对 DSM302 来说，参照 90-30 系列 PLC 用户手册中 GFK 1464中关于 DSM 302，对 DSM314 来说，参照 90-30 系列 PLC 用户手册中 GFK 1742 中关于 DSM 314 。

注意：只有当处理器为 350 ， 352 ， 360 ， 363 ， 364 和 374 ，并且版本为 10.00 或更高的版本时候， DMS314 才有效。

表 2-3. 在90-3-0 35x,36x,37x系列CPU中，I/O扫描时间（毫秒）

模块类型		CPU Model						
			331			340/341		
		311/313 /323	主机架	扩展机架	远程机架	主机架	扩展机架	远程机架
8位离散输入		.076	.054	.095	.255	.048	.089	.249
16位离散输入		.075	.055	.097	.257	.048	.091	.250
32位离散输入		.094	.094	.126	.335	.073	.115	.321
8位离散输入		.084	.059	.097	.252	.053	.090	.246
16位离散输入		.083	.061	.097	.253	.054	.090	.248
32位离散输入		.109	.075	.129	.333	.079	.114	.320
8位混合输入/输出		.165	.141	.218	.529	.098	.176	.489
4通道模拟量输入		.151	.132	.183	.490	.117	.160	.462
2通道模拟量输出		.161	.138	.182	.428	.099	.148	.392
高速计数器		2.070	2.190	2.868	5.587	1.580	2.175	4.897
电源辅助转换管理 (1-轴)		2.330	2.460	3.175	6.647	1.750	2.506	5.899
电源辅助转换管理 (2-轴)		3.181	3.647	4.497	9.303	2.154	3.097	7.729
DSM 302*	40 AI, 6 AQ	3.613	4.081	5.239	11.430	2.552	3.648	9.697
	50AI, 9 AQ	4.127	4.611	5.899	13.310	2.911	4.170	11.406
	64 AI, 12 AQ	4.715	5.276	6.759	15.747	3.354	4.840	13.615
GCM	无设备	.041	.054	.063	.128	.038	.048	.085
	8 64位设备	11.420	11.570	13.247	21.288	9.536	10.648	19.485
GCM+	无设备	.887	.967	1.164	1.920	.666	.901	1.626
	8 64位设备	4.120	6.250	8.529	21.352	5.043	7.146	20.052
PCM 311	未配置,或无应用任务	N/A	3.350	N/A	N/A	1.684	N/A	N/A
	尽可能快读 128 %R	N/A	4.900	N/A	N/A	2.052	N/A	N/A
ADC 311		N/A	3.340	N/A	N/A	1.678	N/A	N/A
16通道模拟量输入 (电流或电压)		1.370	1.450	1.937	4.186	1.092	1.570	3.796
I/O 主联接	无设备	1.910	2.030	1.169	1.925	.678	.904	1.628
	16 64-位设备	6.020	6.170	8.399	21.291	4.992	6.985	20.010
I/O 从联接	32-位	206	.222	.289	.689	.146	.226	.636
	64-位	.331	.350	.409	1.009	.244	.321	.926

*对于应用程序，DSM对扫描时间的影响将影响到机器的运行你需要用DO I/O功能块。延迟I／O和快速访问背板总线服务请求发送和获得必要的在每次扫描过程中没有获得所有数据。对 DSM302 来说，参照 90-30 系列 PLC 用户手册中 GFK 1464中关于 DSM 302，对 DSM314 来说，参照 90-30 系列 PLC 用户手册中 GFK 1742 中关于 DSM 314.

注意:311 341 CPU 不支持 DSM314

扫描时间计算

表 2-1 列出PLC 扫描时间七个步骤。扫描时间由固定时间（内务处理和诊断）和可变时间组成。 可变时间根据I／O配置，程序的大小和连接到PLC的编程设备类型的变化而变化.

关于扫描时间计算的例子

下表说明了在 90-30 系列中 311 PLC 扫描时间的计算

用于计算的模块和指令在下表中列出:

? 输入模块: 5 个 16–位的 90–30 系列输入模块.

? 输出模块: 4 个 16–位的 90–30 系列输出模块.

? 程序指令: 700 个布尔变量.(或，与,等等)300 个输出线圈和 200 个数学功能(加,减等等).

扫描	计算	时间		
		无编程器	HHP	逻辑
系统初始化	0.705ms	0.705ms	0.705ms	0.705ms
数据输入	0.055 x 5 = 0.275ms	0.275ms	0.275ms	0.275ms
执行程序	1000 x 0.4∝s* + 200 x 89∝s** + 18.2ms	18.2ms	18.2ms	18.2ms
数据输出	0.061 x 4 = 0.244ms	0.244ms	0.244ms	0.244ms
编程器服务	0.4ms + programmer time + 0.6ms	0ms	4.524ms	2.454ms
无编程器服务	无	0ms	0ms	0ms
重新配置	0.639ms	0.639ms	0.639ms	0.639ms
诊断	0.048ms	0.048ms	0.048ms	0.048ms
PLC 扫描时间	系统初始化+ 数据输入 +执行程序 + 数据输出+编程器服务+无编程器服务+诊断	12.611ms	17.135ms	15.065ms

PLC 扫描步骤:

这一节讨论关于PLC扫描主要过程的几个步骤:

1. 内务管理
2. 输入扫描
3. 应用程序逻辑扫描
4. 输入扫描
5. 编程器服务
6. 系统通讯
7. 重新配置
8. 校验和计算

1. 内务管理

内务管理是为扫描开始做的一切的准备工作.假如PLC工作在标准扫描模式,扫描将继续直到需要的扫描时间结束, 假如需要的扫描时间没有结束,The OV-SWP%SA002被设置,扫描将继续,另外,定时器的值(0.01s,0.1s和1s)将被从通过计算第一次扫描的值和最近一次的扫描值而更新.为了保持积累,目前扫描开始都是以100毫秒为增量单位.每一个定时器都有一个包含100毫秒增量的.当上一个定时器的值增加的时候它就开始计时了.

2. 输入扫描

输入的扫描在扫描的输入扫描部分发生,它优先于程序的执行。 在扫描这部份期间, 所有系列90-30 输入模块被扫描, 而且他们的数据存储到%I离散量输入或%AI模拟量输入中。Ginius通讯模块GCM,增强Ginius通讯模块GCM+, 或Ginius总线控制器接收的全局数据存储到%G中。按参考地址升序顺序扫描模块, 由Ginius模块开始, 然后是离散量输入模块, 最后是模拟量输入模块。
如果CPU是在停止模式而且CPU配置成无I/O扫描 ,输入扫描将被跳过。

3. 应用程序逻辑扫描

当输入扫描完成后, 应用程序逻辑扫描就立刻执行了。 应用程序逻辑扫描的有两个主要任务:(1)处理/ 执行程序逻辑 (2)更新 %Q,%AI, 和 %AQ 输出。 (然而, 直到输出扫描时, 输出模块才被更新).通常情况下,梯形逻辑图是从左到右, 从上到下来执行的。虽然流程能够被调用和跳转临时改变。

但是逻辑程序只有当遇到END或者是缺省程序逻辑的结束标志后才会停止。

313 和更高级别的CPU有一个布尔指令处理器 , 80C188，80386或者AMD SC520微处理器执行定时器，计数器和功能块函数。 在型号311 和 90-20CPU中， 80 C188 执行所有的布尔变量，定时器，计数器和功能块指令。 在Micro中，H8 处理器运行所有的布尔变量和功能块函数。
.

指令执行时间参见附录A。

4. 输出扫描

程序执行以后是输出扫描部分。输出将根据％Q数字量输出和％AQ模拟量输出数据更新。假如你有一个Genius通讯模块或者是Genius总线控制器用于传出全局数据,然后数据从 %G 内存中被发送到 GCM,GCM+,或GBC。系列90-20和Micro输出扫描只包括离散输出。 在输出扫描的时候，所有的90-30系列的输出模块按地址增加顺序扫描。所有的输出数据被发送到90-30系列输出模块时，输出扫描完成。

如果CPU是停止模式， 并且CPU配置中的IPScan-stop参数被设置成NO,输出扫描将被跳过.

警告

假如CPU配置中IPScan-stop参数被设置成YES,即使在PLC在停止模式，real-world输出为ON，因为在输出模式中，PLC将把当前值写到输出模块。

.

5. 编程器通讯窗口

这部分扫描的主要是和编程器进行通讯，假如有一个编程器被检测到，CPU将运行编程器通讯窗口。假如没有编程器被检测到或者系统中的没有配置模块，编程器通讯窗口将不运行。每次扫描配置一个模块。

支持手持编程器以及可以连接到串口和用SNP协议的其他编程器，支持编程器和智能模块通讯。

编程器通讯窗口模式

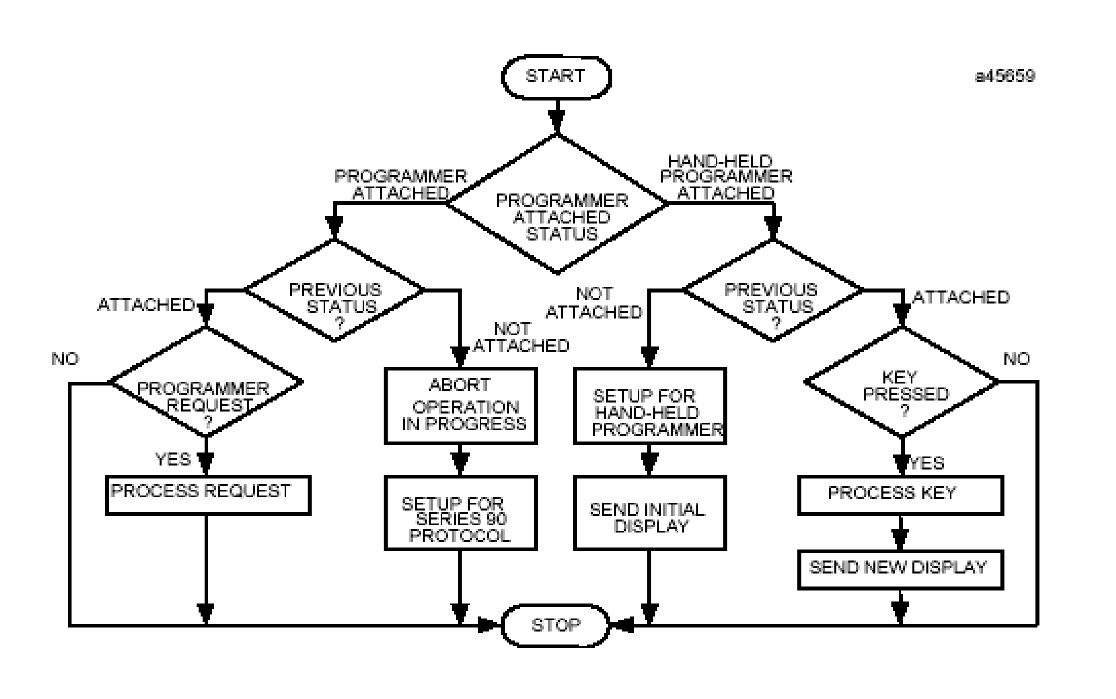
- **限制模式**. 在默认的限制窗口模式中，每次扫描对于编程器CPU将执行一个动作，也就是说，对于每一次按键，它提供一个服务请求或回应。 如果编程器请求需要超过6毫秒(或8毫秒由CPU决定，见注释)，请求处理过程将被分到几个扫描周期执行。

注意

对340和更高的CPU来说，通讯窗口的时间限制为6毫秒，对311，313，323和331模块中，通讯窗口的时间限制为8毫秒.

- **完全模式.** 在完成模式中, CPU将管理编程器通讯直到通讯完成或者直到50毫秒。.

下面这个图是编程器扫描通讯的流程图.



图示 2-2. 编程器通讯窗口流程图

6. 系统通讯窗口 （适用于331和更高CPU）

这是来自智能通讯模块中的通讯请求的流程（见流程图），例如PCM或者DSM。请求建立在请求先到服务先到的基础上，然而，当智能模块进入循环状态时候，就没有先后顺序了。.

在默认的RrunTto-Completion模式中，系统通讯窗口的长度限制在50毫秒内.假如智能模块请求需要时间超过50毫秒，请求将被分到几个扫描周期因此没有一次扫描被影响到。

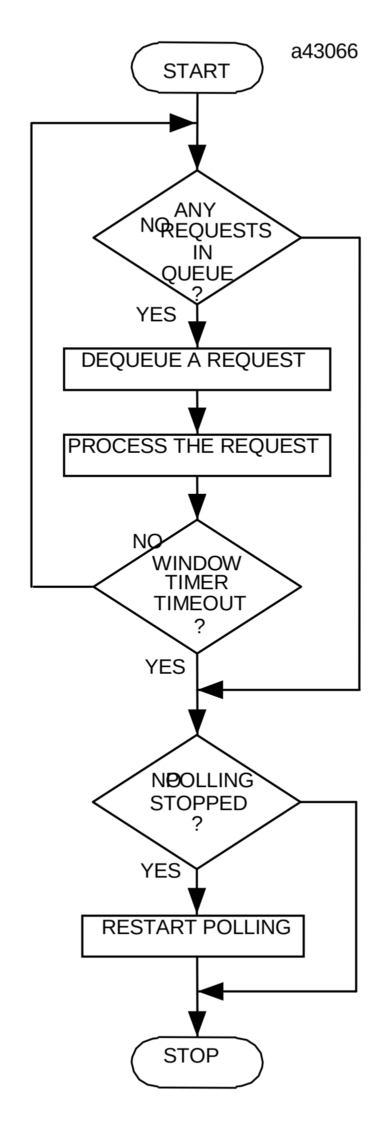


图 2-3. 系统通讯流程图

7. 重新配置

在扫描这部分的时候，cpu检查实际的硬件内容与配置的硬件内容是否相同。为一个模块配置的插槽，但物理上是空的，或者插槽上接了有故障的模块，插槽将不被cpu扫描。(例如：cpu将不从模块或插槽读任何输入数据，也不会向模块或插槽发送任何输出数据). 在重新配置中,如果cpu检测到先前被鉴别到安装了有故障模块的插槽现在换成了好的模块,或者被配置的模块物理上也被加到plc上,cpu将开始扫描这个模块.重新配置可使cpu做到下面各项：

- 确认配置改变有效。
- 忽略来自故障模块或缺失模块的不准确的输入数据。
- 避免有故障的输出模块发送输出数据冲突。

8. 校验和

校验和计算在扫描周期用户程序执行时计算.因为它将需要很长时间对整个程序进行校验和计算，你可以在cpu的配置屏幕上指定从0到32个字被检查。

如果计算的校验和与校验和变量不匹配,程序报校验和故障。这将记录到PLC故障表中，plc的模式将转为stop.如果校验和计算失败，编程器通讯窗口不受影响。缺省校验和字数是8。

PCM 与plc的通讯 (Models 331 and Higher)

对于智能选项模块(IOM)，例如pcm，当他们需要服务时，没有办法去中断CPU. CPU一定轮流检测（周期性的检测）每个智能选项模块的服务请求。在扫描过程中这种轮流检测不会同时发生（看下面系统流程图）。

当智能选项模块已被登记并向CPU发送服务请求，这个请求将会在系统通讯窗口排队进行处理。

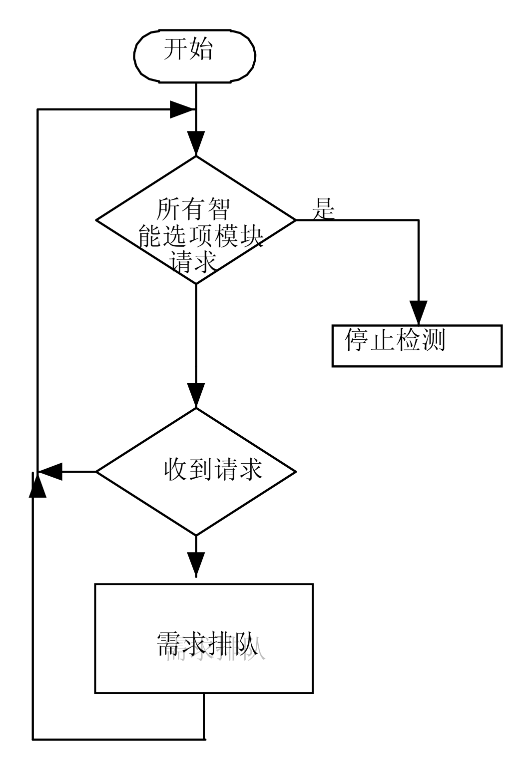


图 2-4. PCM 与PLC通讯

数字伺服模块 (DSM)与PLC通讯

DSM302 and DSM314是不同时于90-30系列CPU模块运行的智能选项模块 .数据经过%Q, %I, %AQ,和%AI存储器在CPU与DSM之间进行交换。一个PLC的CPU需要时间通过PLC底板和DSM模块进行读写数据的交换。表格2-2列出了对于各种可能的DSM配置的扫描影响。
对于应用DSM模块其他时序，参考下列手册：

- *Motion Mate DSM302 for系列s 90-30 PLCs 用户手册*, GFK-1464.
- *Motion Mate DSM314 for系列 90-30 PLCs 用户手册*, GFK-1742.

标准程序扫描

除了标准程序扫描正常执行，会遇到一些变化。在以下描述这些变化,可以从编程软件中显示和/或改变。

恒定扫描时间模式

在标准程序扫描中，每一个扫描执行尽可能快的运行。恒定扫描时间模式，这种扫描模式下，每次扫描时间恒定。你可以设置恒定扫描，将变成默认的扫描模式，每一次PLC从STOP模式到RUN模式采用这种扫描模式。你可以设置恒定扫描时间模式的时间，对于311-341CPU为5-200毫秒，对于350-364和374CPU为5-500毫秒。

由于PLC各扫描步骤所需要的时间是变化的，恒定扫描时间应该比正常扫描模式下扫描时间多10毫秒，这可以避免扫描超时的故障。

使用恒定扫描时间模式，当I/O点或寄存器的值循环使用，例如在控制运算法则中。另一个原因可以被确认，在输出扫描和下一次输入扫描之间，从程序接收输出数据后允许处理输入。

如果在扫描完成之前，恒定扫描定时器到，整个扫描，包括通讯窗口完成。然而，扫描超时故障将在下一个扫描周期开始时产生。

配置恒定扫描模式

有两个方法去配置恒定扫描模式：

- 在Logicmaster配置软件中，CPU配置中可配置扫描模式和扫描时间参数。在你做出选择之后，你必须在停止模式下，将配置的参数存储到PLC中。一旦存储进去，这个配置将变成默认的扫描模式。

- Logicmaster编程软件中,在PLC的控制和状态菜单上，PLC扫描选择栏中有扫描模式和时间参数选项。这个屏幕上的参数只有在运行模式下才可以被编辑。在屏幕上改变模式只能存储在PLC上，不能被存储在电脑的文件夹中，而且只有在下一次PLC进入到运行模式时才能生效。对于暂时配置的扫描模式，这种方式对于系统设计和调试运行时非常有用的。

在停止模式下PLC的扫描

当PLC进入到停止模式下，应用程序不被执行。与编程器和智能选项模块继续通讯。此外，当进入停止模式，故障模块发送请求和模块重新配置继续执行。为了效率更高，运行系统使用更大的“时间分割“值高于用于运行模式的值（通常大约50毫秒每个窗口）。你可以选择I/0是否被扫描。如果CPU的屏幕上的I/OSCAN-STOP参数被设置为YES，那么在停止模式下，I/0扫描可以执行。

注意

如果在CPU屏幕上，IPScan-Stop的参数被设置成YES，即使PLC在停止模式下，实际的输出可以被打开，因为在当前输出扫描期间PLC将会把当前值写到输出模块中。

通讯窗口模式

对于编程器通讯窗口，默认窗口模式是一种“限制”的模式。那意味着如果一个请求运行超过6毫秒，这个请求的运行分到多个扫描周期，所以没有一个扫描周期超过6毫秒。对于313，323和331CPUS在运行模式时，扫描周期可以是12毫秒。在Logicmaster上使用“扫描控制”窗口可以改变激活窗口模式，对于如何改变激活窗口模式，参考*Logicmaster 90™ Series 90™-30/20的编程软件用户手册*第5章“PLC控制和状态”。

注意

如果系统窗口的模式被变成限制。像PCM和GBC这样的选项模块使用系统窗口与PLC通讯将对扫描时间产生很少的影响，但是对请求的反应将会变慢。

35x, 36x和37x系列CPU钥匙开关作用:改变模式和闪存保护

对于所有350—374 CPUs(CPUs 311-341 除外)都有一个钥匙开关，然而，有一些版本的PLC不是支持钥匙开关的所有特性。这些不同将在这节中讨论。注意一些PLC上的钥匙开关标注着开/运行和关/停止，而一些PLC上的钥匙开关则标注着开和关。无论标注着什么,所有的钥匙开关都象下面所描述的工作。

闪存保护（(Hard-Wired)

这个 hard-wired具有不可配置的特性，可以被用于阻止闪存被没有授权的人（没有钥匙的人）改动，当钥匙开关处于开位置的时候，闪存不能被写入。只有钥匙开关处于关位置的时候，闪存才能被写入。无论下面两个配置特性怎样被设置，钥匙开关的这个特性总是生效的。

运行/停止(可配置的)

这个可配置的特性在CPU固件7.00版本中介绍。这个特性可以通过CPU的配置窗口上的**R/S**开关的参数进行设置。**R/S**开关的参数默认为不使能。如果**R/S**开关的参数被设置为使能，你可以将钥匙开关旋转到关状态来停止PLC,将钥匙开关旋转到开启状态来启动PLC（如果没有故障的情况下）。如果出现故障，下面之一将会发生：

- **I如果PLC不是致命故障，**将钥匙开关从OFF状态旋转到ON状态，将会使PLC进入到运行状态。运行指示灯将亮着，但是故障表不会被清除。

- **I如果PLC发生致命故障,** t将钥匙开关从OFF状态旋转到ON状态将会使运行指示灯闪烁五秒钟，PLC不会进入到运行状态。这个灯闪烁表示故障表中有一个或更多个致命故障。你可以再次将钥匙开关从OFF状态旋转到ON状态，在五秒的时间内，清除故障表中的故障（如果五秒的时间到了，你可以将钥匙开关从OFF状态旋转到ON状态，将启动下一个五秒种）。如果是用这种方式不能清除掉故障，在恢复运行之前，你不得不修复发生致命错误的原因。故障的细节问题请看第3章。

其他的运行/停止键锁选择：

- 如果**R/S 开关**参数被设置为使能状态，钥匙锁开关处于OFF的位置，PLC将进入到停止模式，PLC软件将不能用于将PLC设置成运行模式。

  如 果 **R/S**开关参数被设置为使能状态，钥匙锁开关处于ON的位置，并且PLC没有致命故障，可以用编程软件转换PLC的运行模式和停止模式。

- If如果**R/S**开关参数被设置为使能状态，钥匙锁开关在ON的位置，但是PLC是停止着的，你可以通过将钥匙开关从OFF位置旋转到ON位置使PLC进入到运行状态，或者通过编程软件来实现。

内存和强制保护（可配置的）

这个特性在8.00CPU版本中介绍的。它可以通过CPU配置画面上的内存保护参数来设置。内存保护参数默认状态下是不起作用的。如果内存保护参数被设置为使能状态，钥匙开关在ON位置，下面所描述是真的：
I

- 用户内存 (编程和配置)不能被改动。
- 离散点不能强制。
- 日历时钟不能通过手持编程器进行改动（但是可以使用配置软件来改动日历时钟TOD）

保护好钥匙

每个新的350—374型号CPU都配备了钥匙开关。如果你使用上面所描述的钥匙开关的一个或多个特性，我们建议你要保管好你的钥匙。如果钥匙丢了、安错了地方或者被偷了，你就不能在你的PLC上工作，而且没有被授权的人也可以进去你PLC内部。如果想恢复或者不止两个人想进入PLC内部，你可以购买备用的钥匙。一套钥匙锁和钥匙，包括3把钥匙。可以从GE分销商处购买。当你订购的时候，需要目录号码44A736756-G01.。所有350—374型号PLC使用的都是一样的钥匙

关闭钥匙开关的特性

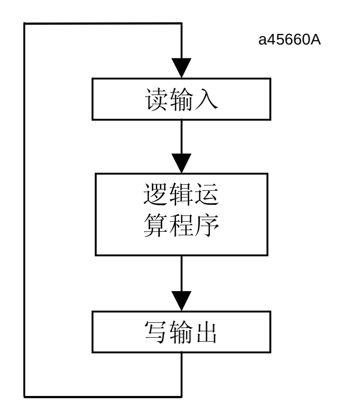
如果你不需要使用钥匙开关的任何保护特性，你可以选择使他们不起作用。将钥匙开关置于关闭位置，将钥匙拔下，将**R/S** 开关参数和内存保护参数（上面所描述）设置为不起作用状态（默认设置），就可以实现。在这种条件下，所有钥匙开关的保护特性都不起作用，你也不需要使用钥匙进入PLC内部。

第2节：程序组织和用户参考书目/数据

90-30系列编程控制器的用户内存大小显示在下列表格中.

用户内存大小	
CPU 模块	**用户内存 (千字节)**
CPU311	6
CPU313, CPU323	12
CPU331	16
CPU340	32
CPU341	80
CPU350	80 (9.00版本和更高版本) 32 (低于9.00的版本)
CPU351, CPU352, CPU360, CPU363, CPU364, CPU374	240 (9.00版本和更高版本) 80 (低于9.00的版本)

从9。00版本的CPU开始，对于351, 352,，360, 363, 364和374型号CPU的%R, %AI, 和%AQ的内存大小是可以配置的。（详细情况，*90™*系列的逻辑控制器的*90™-30/20编程软件的用户手册*，GFK-0466K，或者*编程软件的用户手册）*。对于90-20系列的编程控制器的211CPU模块，一个程序可以到2 KB，允许每个程序块（主程序快或子程序块）最大数量为3000。对于90-30系列PLC，C程序块最大尺寸可到达80KB，LD程序块和SFC程序块为16KB;然而对于SFC功能块，16KB中有一部分要为内部数据块使用。就象下面表格所标示的那样，当PLC在正常运行模式时，用户的程序被PLC重复的执行。



参考*90-30系列编程控制器的安装和硬件手册*，GFK-0356，或者*90-20系列编程控制器的用户手册*，GFK-0551，是描述每个CPU模块的程序大小和参考变量的限制的。所有程序都有变量表，列出了所有变量和相关描述，变量的被分配到用户的程序中。程序块的声明列出了子程序块与主程序的关系。

.

子程序块

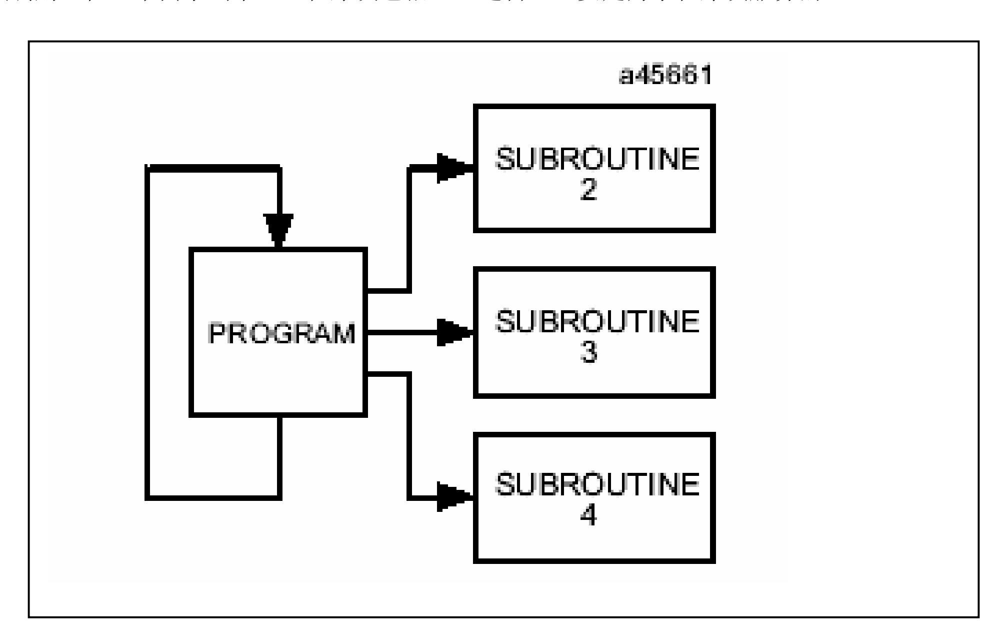
一个程序可以调用子程序块。对在子程序使用调用指令之前，子程序必须经过程序块编辑器定义。一个程序中最多可以定义64个子程序块，程序中的逻辑程序块最多可以使用64条调用指令。一个子程序最大16KB或者3000条，但是主程序和所有的子程序块的大小不能超过使用CPU模块限制的。

注 意

子程序块不支持90-20系列的PLC或者微型PLC.子程序的使用是可选择的。将一个程序序分成多个子程序块可以简化程序，提高控制运算的可读性，可能减少程序必须的逻辑运算。

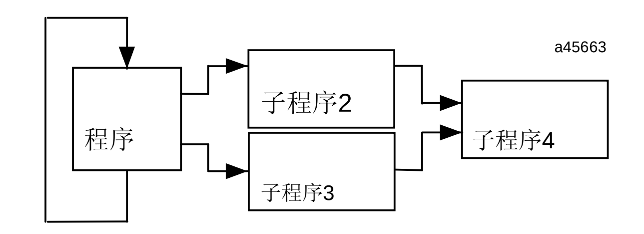
如何使用子程序块举例

作为一个例子，一个程序的逻辑可以被分成三个子程序块，每个子程序块可以在程序需要的地方被调用。在这个例子当中，主程序块包括一些逻辑，主要是为子程序块服务的。



在程序运行时，一个子程序块可以被使用很多次。程序中一个逻辑需要重复多次，那我们就可以将这个逻辑做成子程序块。我们可以通过调用子程序执行程序。用这种方法，可以减少整个程序的大小。

此外在程序中调用，子程序块也可以被其他子程序块调用（这称为嵌套）。子程序块甚至可以被自己所调用。



在“应用程序堆栈溢出”故障出现，PLC转换到**停止/故障**模式之前，PLC只允许有8层嵌套。主程序是调用嵌套的第一级。

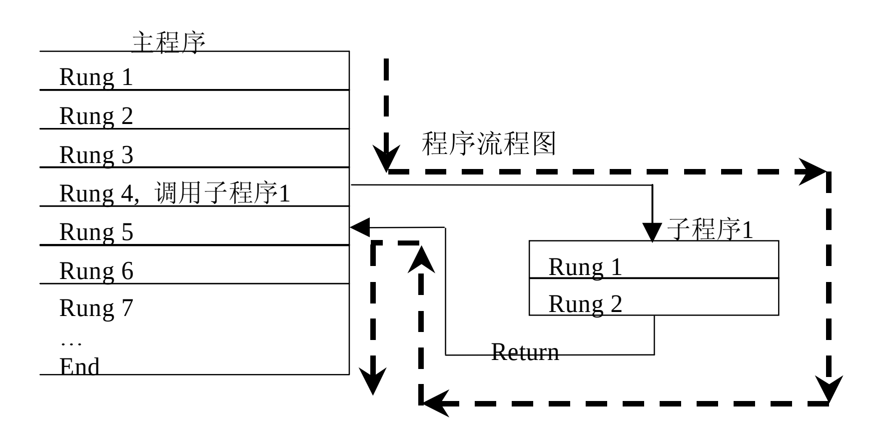
如何调用程序块

在梯形图程序中或其他子程序块中，一个子程序块被调用运行。

```
|
|%I0004                                                         %T0001
|——| |————————————————————————————————————————————————————————————————( )—
|            _______________
|%I0006     |               |
|——| |——————|  CALL ASTRO   |—
|           | (SUBROUTINE)  |
|           |_______________|
|
|%I0003  %I0010                                                  %Q0010
|——| |——————| |———————————————————————————————————————————————————————( )—
|
```

包括子程序在内的程序运行结果

如果一个子程序在主程序或其他子程序中被调用，被调用的子程序将会运行到结束，然后将所控制的结果返回给调用它的主程序或子程序。所控制的结果将会返回到子程序调用的下一个句子。在下面的例子中，深色虚线中表示的是程序流程图（是按逻辑顺序执行的）。在这个例子当中，一个简单的双语句子程序从主程序的第四句开始被调用。这个双语句的子程序被执行之后，程序流程图返回到主程序，并从第5句开始。



周期性子程序

340型或更高型号的CPU的4.20版本或更新版本支持周期性子程序。请注意下面的限制。

1. 在周期性子程序中，定时器 (TMR, ONDTR, 和 OFDTR)功能块不会完全执行。周期性子程序中的DOIO功能块的参考范围包括被分配到智能型输入/输出模块(HSC, APM, DSM, Genius等.)将会引起CPU与这个模块失去通信。在周期性子程序运行时，FST_SCN 和LST_SCN联系在一起(%S1和%S2)将会生成一个不确定的值。一个周期性子程序不能调用或者被其他子程序所调用。

2. 如果主站上没有PCM, CMM, 和ADC模块，周期性子程序的反应时间为0.35毫秒（也就是，周期性程序应该运行的时间和实际运行时间的间隔。如果在主站上有PCM, CMM或ADC模块，即使没有被配置或使用，反应时间大约为2.25毫秒。由于这个原因，对于PCM产品，周期性子程序不见以被使用。

用户参考变量表

应用程序的数据被存储在寄存器或参考变量表中。

表2-4. 寄存器参考变量

类型	描述
%R	%R用于分配系统寄存器的参考变量，存储计算结果之类的程序数据。
%AI	%AI表示一个模拟量输入寄存器。他后面跟随的是和寄存器的相关地址（例如，%AI0015）。模拟量输入寄存器保存着一个模拟量的输入值或其他值。
%AQ	%AQ 表示一个模拟量输出寄存器。他后面跟随的是和寄存器的相关地址（例如，%AQ0056）. 模拟量输出寄存器保存着一个模拟量的输出值或其他值。

注意：
所有的寄存器通过重新上电保存到CPU中

表2-5 离散参考地址

类型	描述
%I	%I前缀表示输入参考地址。在输入表格里前缀后面紧跟的参考地址（例如%I00121）。%I参考在最后一次输入扫描的时候，被载入这些来自输入模块的储存所有标准输入状态的输入状态表。参考地址通过配置软件或手动程序被赋值给离散输入模块。在参考地址被赋值前，没有来自模块的数据被接收。%I数据是可保持的或不可保持的 .
%Q	%Q 前缀描述的是输出参考地址。线圈核对逻辑控制功能 90-30/20/微型的软件控制为成倍使用%Q 伴随继电器线圈或输出功能。由软件版本 3 开始，你能选择希望被核对的线圈的水平（单一，警告性复用，复用）。关于特征的更多的信息，查看编程软件使用者手册，GFK-0466。 在输出表格里，%Q前缀后面紧跟的是参考地址（例如：%Q00016）。%Q参考在最后被设置的时候，通过应用程序被载入储存输出参考状态的输出状态表。通过输出扫描，这个输出状态表的值被送入输出模块。参考地址通过使用配置软件或程序员手动操作，被赋值给离散输出模块。特殊的%Q参考可以是可保持的或非可保持的。
%M	%M前缀描述的是内部参考地址。线圈核查功能通过继电器线圈或输出功能核查了%M参考的复用。你可以选择被希望的线圈核查水平（单一，警告复用，复用）。更多特征信息请参考GFK-0466。特殊%M参考可以是可保持的或非可保持的。
%T	%T前缀描述了暂时性参考地址。因为那些参考不能为多重线圈使用核查，他们能在同一程序中使用多次，甚至在线圈使用效验被激活的时候。当使用剪切/粘贴和文件书写/包含功能时，%T能够被用来防止线圈使用冲突。因为储存器被分给暂时性使用，它在功率损耗或RUN-TO-STOP-TO-RUN转换中是不被保留的，而且不能和可保持性的线圈使用。
%S	%S前缀描述的是系统状态参考。那些参考被用来访问特殊PLC数据，例如定时器，扫描信息和差错信息。系统参考包括%S，%SA，%SB，和%SC参考。%S，%SA，%SB，和%SC被用于一些联系。%SA，%SB，和%SC被用于可保持性的线圈。%S能被用作字或位串输入论据给函数或函数块。%SA，%SB，和%SC能被用作字或位串输入输出给函数或函数块。
%G	%G前缀描述全局数据参考。那些参考地址在几个PLC间被用作访问数据共享。%G参考可被用于联系的可保持的线圈。因为%G存储器是可一直被保持的。%G不能用于非可保持性线圈。

保持性是基于线圈的类型，有关“保持性数据”更多信息在下一页。

别名：

使用者可能随意的给一个参考地址取个别名。别名是非常有用的，因为它能够给使用者传达关于用途或函数地址的信息。例如，在安置于工厂的一个 PLC 系统，输出线圈%Q0001 被用来给一个控制物理泵的电动器启动延迟器加电压，一般被工厂员工称为“1 号泵”。把别名 1 号泵赋给%Q0001 将有助于维修系统故障的员工认识%Q0001 的用途。

别名必须以字母和开头，大概 1-7 个字符长。对存储器地址（参考）和别名的区别，百分号%被用作存储器的第一个字符。那么，例如，M1 被考虑作为 PLC 中的一个别名，但%M1 则被考虑作为存储器地址。关于别名的更多的信息，请参看手册 GFK-0466（90-30PLC 系列逻辑控制使用者手册）

跳变和强制

%I，%Q，%M 和%G 用户变量有跳变位和强制位。%T，%S，%SA，%SB 和%SC 只有跳变位，没有强制位。CPU 使用跳变位给计数器和跳变线圈。注意计数器不能使用同中跳变位作为线圈。计数器跳变位被储存于内部参考地址中。

在 331 型或更高型的 CPU，强制位能被设置。当强制位被设置，相关参考地址不能从程序或输入装置改变，只能通过程序员命令改变。CPU323 型，321 型，313 型和 311 型，以及微 CPU 都不支持强制变量。

数据保持性

当 PLC 停止时，数据能保存，则称为可保持的。90 系列 PLC 保存程序逻辑，故障表和诊断，强制和输出强制，字数据（%R，%AI，%AQ），位数据（%I，%SC，%G，故障位和保留位），%Q 和%M 数据（除非使用非保持性线圈），和存储于%Q 和%M 里的字数据。%T 数据是不被保存的。通过以上描述，虽然%SC 位数据是可被保持的，但%S，%SA 和%SB 默认的是非保持的。

当无论什么时候使用非保持线圈时，%Q 和%M 参考是非保持的（意思是，当 PLC 开关从 STOP 打到 RUN 时，将被清除）。非保持线圈包括线圈—( )—，否定线圈—(/)—

，置位线圈—(S)—，复位线圈—(R)—。当%Q或%M参考用于保持线圈，或用作函数块输出，内容可保持即使调电或从运行到停止模式。保持线圈包括保持线圈—( )—，否定保持线圈—(/M)—，保持置位线圈—(SM)—，保持复位线圈—(RM)—。最后一次%Q或%M被编程线圈指令决定线圈类型%Q或%M是保持的或非保持的。例如，如果Q0001是最后一次编程用于保持性线圈，%Q0001数据将被保持。然而如果%Q0001最后一次编程用于非保持性线圈，%Q0001数据将是非保持的。

数据类型：

表 2—6，数据类型

类型	名称	描述	数据格式
INT Signed	带符号整型	带符号整型使用16位存储器数据区域，用二进制补码表示（第16位是符号位）有效位是-32768~+32767	寄存器 1 S| 16 1 (16 位 位置)
DINT	双精度带符号整型	双精度带符号整型储存于32位数据存储器数据区域（事实上是两个连贯的16位数据区域），用二进制补码表示。32位是符号位。有效范围是-2147483648~+2147483647	寄存器2 S| 32 17 寄存器1 16 1
BIT	位	位数据类型是存储器最小单元。它有两个状态，0和1，一个位串长度为N	(Two’s Complement Value)
BYTE	字节	字节数据类型有一个8位值。有效范围是0~255（在16进制里是0到FF）	
WORD	字	字数据类型用16个连贯位数据存储器代替数据区域位保存一个数。每个位保存它的二进制状态（1或0），这个位串被看做一个整数。有效范围是0到FFFF	寄存器1 16 1 (16 bit positions)
DWORD Double	双字节	除了使用32位连贯位在数据存储器中，双字节数据类型有相同的字符作为单字类型。有效范围是0到FFFFFFFF	寄存器2 32 17 寄存器1 16 1
BCD-4	BCD码	BCD码数字使用16位数据存储区域。每个BCD数字使用4位来表示0~9。这个BCD编码合法取值范围是0~9999	(32 位 寄存器1 4 | 3 | 2 | 1 16 13 9 5 1 (4 BCD 码)
REAL	浮点型	浮点数使用32连贯位（事实是2个连贯16为存储区域）。以这个格式的数字存储范围是1.401298E-45到+3。402823E+38	寄存器2 S| 32 17 寄存器1 16 1 17

S=标记位（0=正，1=负）

系统状态变量

90系列PLC系统状态变量被分配给%S，%SA，%SC存储器。他们均有别名。例如秒脉冲触点包括T_10MS,T_100MS,T_SEC和T_MIN。例如FSC_SCN,ALW_ON,和ALW_OFF。

注意

%S 是只读位，不能写入。然而你可以写入到%SA，，%SB，%SC 位。

下面列出的是能被用在应用程序里的系统状态变量。当写入逻辑，参考地址或别名均能被使用。查阅第三章“错误解释和修正”，更多细节故障描述和修正故障的信息。你不能使用那些特别的别名给另外的存储器参考命名。

表2-7 系统状态变量

参考	别名	**定义**
%S0001	FST_SCN	当前扫描是第一个扫描周期时置1
%S0002	LST_SCN	当前扫描是最后一个扫描周期时，复位1到0
%S0003	T_10MS	0.01秒分时触点
%S0004	T_100MS	0.1秒分时触点
%S0005	T_SEC	1秒分时触点.
%S0006	T_MIN	分钟分时触点.
%S0007	ALW_ON	常开.
%S0008	ALW_OFF	常闭.
%S0009	SY_FULL	当PLC故障表满时置位. 当记录的故障清除或故障表被清除时复位
%S0010	IO_FULL	I/O故障表填满时置位. 当记录的故障移去或故障表被清除时复位
%S0011	OVR_PRE	当%I,%M或%G超出范围时置位
%S0013	PRG_CHK	后台程序检查被激活时置位
%S0014	PLC_BAT	CPU电池坏时置位，每个扫描周期更新
%S0017	SNPXACT	SNP-X主机激活连接到CPU端口1
%S0018	SNPX_RD	SNP-X主机从CPU读数据
%S0019	SNPX_WT	SNP-X主机写数据到CPU
%S0020		使用浮点数运算正确，置位ON.当输入非数值,被清除
%S0032		保存用于编程软件
%SA0001	PB_SUM	当应用程序计算的校验和与变量校验和不匹配时置位,如果故障只是暂时的，通过向CPU中下装程序可以镜该离散变量清零，若错误是因为RAM故障引起的，需要更换CPU
%SA0002	OV_SWP	PLC在恒定扫描方式下，检测到的扫描时间比规定时间长置位，否则请零。当PLC从STOP转为RUN时清零。

参考	别名	定义
%SA0003	APL_FLT	当发生应用故障时置位，PLC从STOP转为RUN时清零
%SA0009	CFG_MM	PLC上电或下装硬件配置时如果与实际配置不符时置位，纠正后PLC重新上电时清零
%SA0010	HRD_CPU	检测到CPU硬件故障时置位，更换CPU模块后清零
%SA0011	LOW_BAT	电池低时置位，更换电池PLC重新上电后清零
%SA0014	LOS_IOM	I/O模块停止与CPU通讯时置位，更换模块并重新上电后清零。
%SA0015	LOS_SIO	可选模块停止与CPU通讯时置位，更换模块并重新上电后清零
%SA0019	ADD_IOM	机架中添加I/O模块时置位，重新上电后硬件配置匹配时清零
%SA0020	ADD_SIO	机架中添加可选模块时置位，重新上电后硬件配置匹配时清零
%SA0027	HRD_SIO	检测到可选模块硬件故障时置位，更换模块重新上电后清零
%SA0031	SFT_SIO	可选模块出现不可恢复软件故障时置位，重新上电并且硬件匹配时清零
%SB0010	BAD_RAM	上电时，CPU检测到RAM故障时置位，重新上电后产存无误时清零
%SB0011	BAD_PWD	当密码输入错误时,置位.当PLC故障表清空时,清位
%SB0013	SFT_CPU	CPU检测到不可恢复的软件故障时置位，故障表清除时清零
%SB0014	STOR_ER	当编程器存储故障时置位，顺利完成时清零
%SC0009	ANY_FLT	任何故障发生时置位，故障表中没有故障记录时清零
%SC0010	SY_FLT	任何会记录到故障表中的故障发生置位，故障表中没有故障记录时清零
%SC0011	IO_FLT	任何会记录到I/O故障表中的故障发生时置位，I/O故障表中没有故障记录时清零
%SC0012	SY_PRES	PLC故障表中哪怕只有一条故障记录时置位，PLC故障表中没有故障记录时清零
%SC0013	IO_PRES	I/O故障表中哪怕只有一条故障记录时置位，PLC故障表中没有故障记录时清零
%SC0014	HRD_FLT	硬件故障时置位，两个故障表中都没有错误记录时清零
%SC0015	SFT_FLT	软件故障时置位，两个故障表中都没有故障记录时清零

注意：一些没有列在这里的%S 参考地址保留且不必用于程序逻辑

功能块结构

每一行逻辑梯段是由一个或多个程序指令组成的，这些也许是简单的继电器，或许是复杂功能。

逻辑继电器的格式

编程软件包含很多类型的继电器功能。这些功能提供基本电流和逻辑的控制。本例中包含一个普通的常开继电器触点，和一个反线圈。每一个继电器触点和线圈分别有一个输入一个输出。它们一起提供逻辑电流流过触点或线圈。

每一个继电器触点或线圈必须分配一个参考地址，当选择继电器时输入。对于触点，参考地址代表存储位置。本例中，如果参考地址％I0122为ON，电流流过该继电器触点。

%I0122
-| |-

对于线圈，参考地址代表变量存储位置，控制电流流入线圈。本例中，如果电流流入线圈左侧，参考地址％Q004为ON.

%Q0004
-( )-

编程软件和手持编程器都有线圈检查功能，检查用于功能块的继电器线圈或输出的％Q或％M多次使用。

程序功能块的格式（指令）

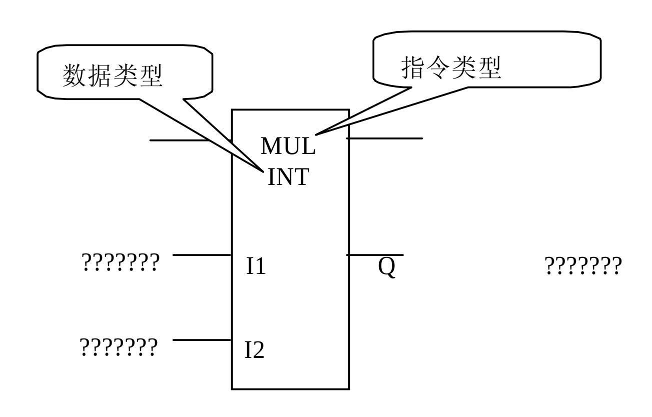
一些功能非常简单，如MCR功能。

-[MCR]-

其它功能复杂一些。他们可能需要很多参数，哪些参数由你输入，是该功能使用的参数。

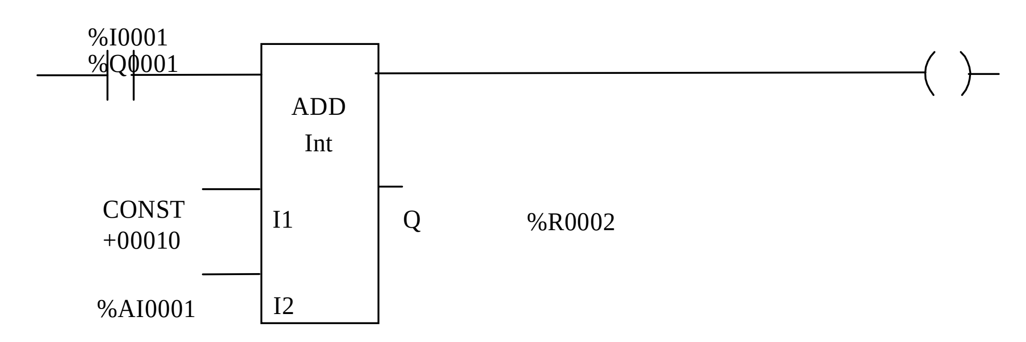
下图功能块示例是乘法MUL指令。然而，参数类型根据功能块类型不同而不同。功能块上部分显示功能名称，本例中还显示数据类型，带符号整数。

很多程序功能（指令）允许你在选择该功能后选择数据类型，例如MUL的数据类型可以是双精度带符号整型D_INT。更多数据类型信息见本章前面描述。



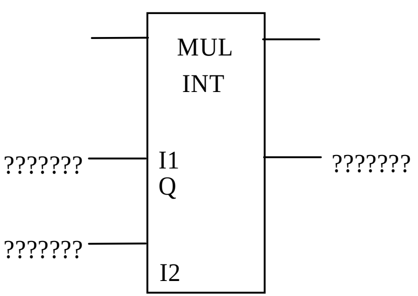
功能块（指令）参数

输入到功能块左侧的每根线代表这个功能块的一个输入，对于这个功能块有两种形式的输入，数字量输入和模拟两输入。数字量输入为通和断两种信号。在下面的画面中，使能触点%I0001就是数字量输入的例子。模拟量输入可以是常量或者是变量。常量是个很明确的值，而变量则是地址。通常如果输入的数据是变化的，我们就使用变量。例如，变量可以是来自模拟测量设备的输入地址。在下面的例子中，对于加法功能块，输入参数I1就是一个常量，输入参数I2就是变量。



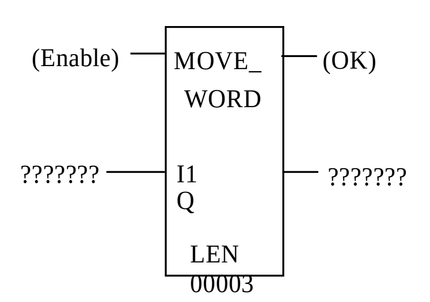
功能快右端的出线表示一个输出。输出可以是数字量，也可以是模拟量。如果是模拟量，这个值被放进一个寄存器中（变量）。在上面的例子中，这个功能块的OK端是数字量，控制输出线圈%Q0001。然而，Q输出端的值是从数学运算得到的结果，所以这地方放了一个寄存器，这个例子中是%R0002。

在功能块的左侧，哪里有问号的标记，你就可以在那里输入数据，那个位置变量数据可以被找到，或者代表变量的数据可以被找到。在功能块的右侧，哪里有问号标记，你通常就可以在这里输入和这个功能块相关的输出变量数据，或者输入和这个功能快相关的输出变量的代表数据。

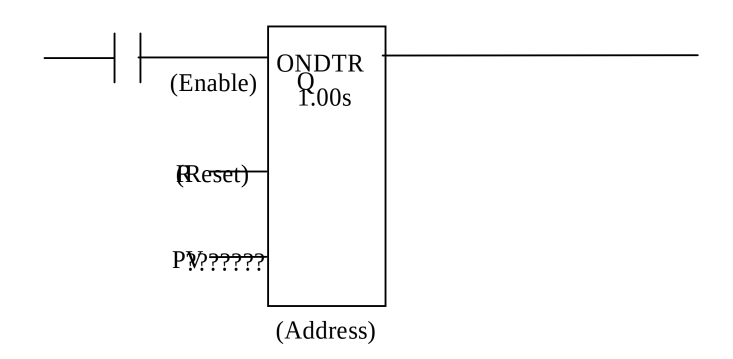


大多数功能块不能改变输入数据，在运算中他们使用输入数据，在输出变量处输出运算的结果。

对于不同的功能块，对一组内存地址（变量）进行运行时，其长度可以选择。在下面的功能块中，LEN 运算功能块可以定义被移动的数量（本例中是3）。

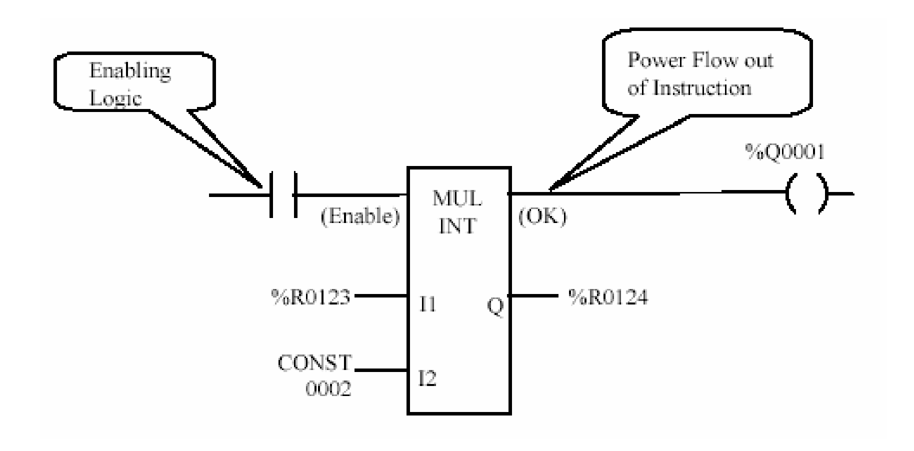


定时器，计数器，BSEQ和ID功能块需要地址来分配3个字（寄存器）来存储当前值，预设值和控制字或者这个功能块的“例子”。这三个连续字的第一个字显示在功能块的下面，用“（地址）”来表示。



功能块电流流入和流出

通过使能输入端使得电流流过功能块。多数功能块都有一个输出，我们称为“OK”输出端。如果功能块完全运行，OK 输出端变为高电平，输出电流。如果有别的设备和OK输出端连接，就向下面所显示的输出线圈，设备可以动了。然而，对于许多功能块OK输出端是可以选择的，因为他们的主要目的是想得到Q输出端的运算（下面的例子是乘法运算）结果的。



注意

假如用logicmaster编程软件，每行语句不能直接以功能块开始，你可以使用S7，ALW-ON（常开）触点.每次扫描都执行该功能块。

电流从功能块右上流出，流向其他逻辑或线圈（可选），功能正确执行流过电流。

第3节：启动方式和停止方式

90-30系列PLC有两种启动方式；冷启动和热启动。CPU通常使用冷启动方式。然而，对于331或更高级别的PLC系统，如果从停电到下一次启动的时间小于五秒钟，通常使用热启动方式。

启动

冷启动方式是由下面下列步骤所组成的。热启动方式跳跃地1个步骤开始执行。

1. CPU将自诊断。自诊断包括检查加了电池之后的RAM，判断RAM中是否有有效数据。
2. 如果有EPROM, EEPROM或闪存，程序中的启动选项指定使用程序的内容，程序的内容要被复制到RAM存储器中，如果没有EPROM, EEPROM或闪存，RAM存储器保持原样，不会复制程序的内容。
3. CPU 访问每一个插槽，判断哪一个插槽上有模块。
4. 硬件配置与软件配置相比较，确保它们是一样的。任何不匹配的配置被检测到，都会被认为是故障，然后报警。例如，如果一个模块在软件中被配置了，但实际硬件中使用的是不同型号的模块，这种条件下就会出现故障，然后报警。
5. 如果软件没有配置，CPU将会使用内置的默认配置。
6. CPU在自身与其他智能模块之间建立通讯。
7. 在启动运行的最后一个步骤中，第一个扫描周期的扫描方式是由CPU的配置所决定的。下一页中的图2-5，显示了当CPU决定是否复制程序时，在停止或运行模式下的启动时，CPU的选择流程图。

注意

上面的第2个步骤到第7个步骤不用于90系列的微型PLC.关于微型PLC的启动和停止的信息,参考90系列的微型PLC用户手册中第5章“系统运行”的“启动方式和停止方式” 部分。

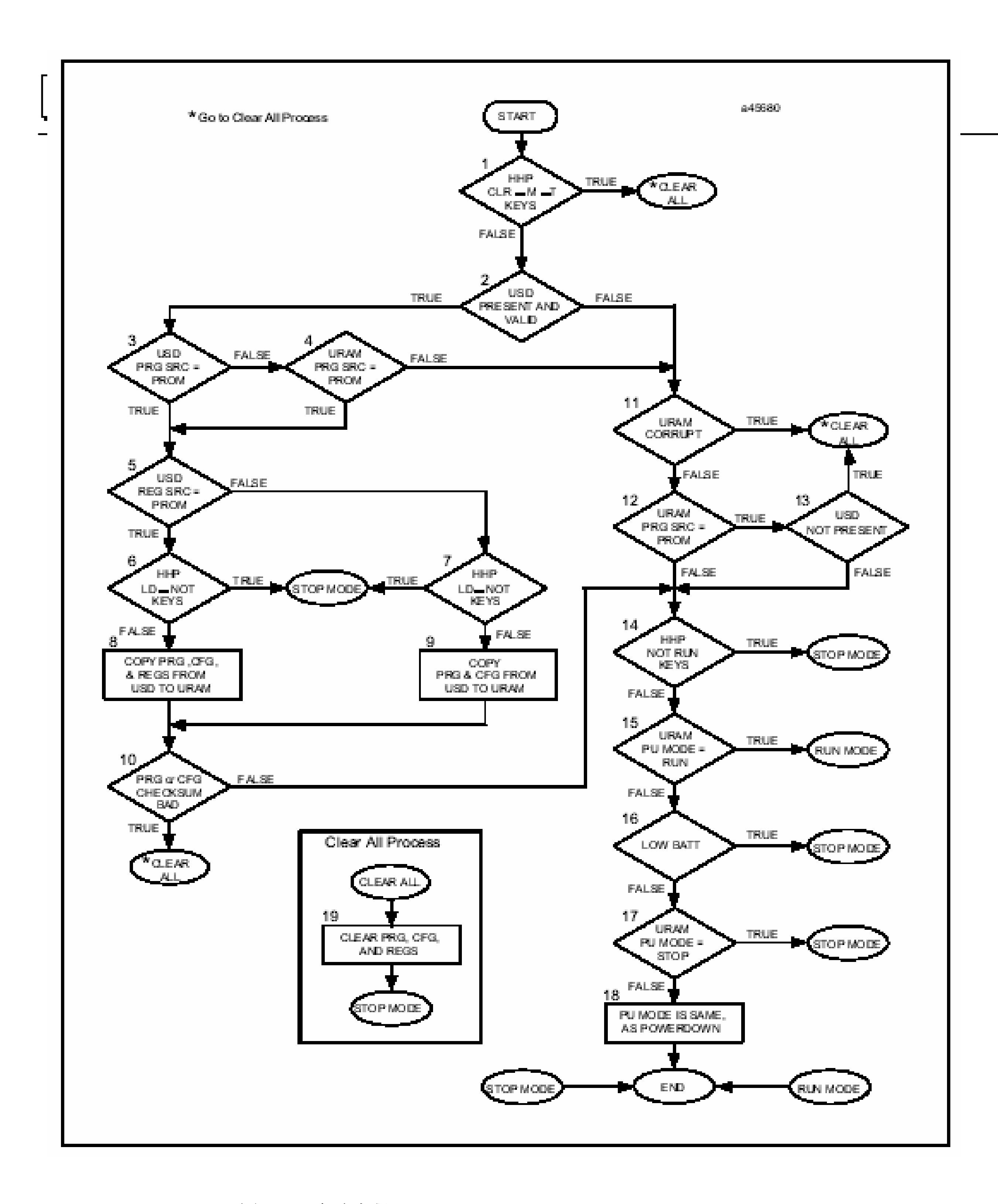


图 2-5. 启动流程

在启动流程图中，开始是最优先的，CPU通过启动诊断，检测CPU使用的各种外围设备和RAM。在完成诊断之后，内部数据结构和CPU所使用的外围设备进行初始化。然后CPU判断用户RAM是否混乱。如果用户RAM混乱，用户程序和配置将被清除掉变成默认的，所有用户寄存器的内容也要被清掉

流程图术语

PRG = User program用户程序 (PRG SRC = Program Source源程序)

CFG = User configuration用户配置

REGS = User registers用户寄存器 (%I, %Q, %M, %G, %R, %AI, and %AQ references).

USD = User storage device, either an EPROM, EEPROM, or Flash device.用户存储设备EPROM, EEPROM或是闪存。

URAM = Non-volatile user ram which contains PRG, CFG, and REGS.包含PRG, CFG, 和REGS的稳态RAM.

HHP = Hand-Help Programmer手持编程器

PU = Power-up启动

CLR = Clear清除

BATT=Battery电池

流程图的扩展文本

(1)在启动到清除URAM过程中，手持编程器上的 <清楚> 和 <M_T>键要一直按着吗？

(2) USD (用户存储设备)存在吗？USD存在有效的信息吗？

(3) USD (用户存储设备)中的PRG SRC参数要被设为Prom含义从USD设备装载PRG和CFG吗？

(4)URAM中PRG SRC参数要被设为Prom含义从USD设备装载PRG和CFG吗？

(5) USD中REG SRC参数要被设为Prom含义从USD设备装载REGS（寄存器）吗？

(6 & 7) 在将PRG, CFG和REGS从USD装载启动的过程中，手持编程器的<LD>和<NOT>键一直按着吗？

(8)将 PRG用户程序, CFG用户配置和 REGS从USD复制到URAM.

(9)将PRG用户程序和CFG用户配置从USD复制到URAM.

(10) PRG和CFG的核对是否只针对非法的USD装载？

(11)检查URAM混乱吗？由于没有电池或电量不足启动，URAM会混乱吗？由于没有没有更新硬件，URAM会混乱吗？

(12) URAM中PRG SRC参数要被设为Prom含义从USD设备装载PRG和CFG吗？

(13) 检查 USD存在吗？这种检查只用于311-341CPU.对于350-364和374CPU，USD是假设存在的。

(14) 在启动到无条件的启动停止模式过程中，手持编程器的<**NOT**>和<**RUN**>键一直按着吗？

(15) URAM中的PWR UP参数要被设置为运行吗？

(16)电量是否低？

(17) URAM中的PWR UP参数要被设置为停止吗？

(18) 无论什么停止模式，都要设置启动模式。

(19) 清除PRG, CFG和REGS.

Note

上一页图中的第一部分不用于90系列的微型PLC。对于微型PLC启动方式和停止方式的信息，参考90系列的微型PLC用户手册中第5章“系统运行”的“启动方式和停止方式”部分。 (GFK-1065).

断电

当电源检测到供电电压过低，或者5伏的直流输出电压低于4.9伏，系统将断电

第四部分　时钟和定时器

在90-30系列PLC中，时钟和定时器包括耗时时钟，日历时钟（对于331，340/341，350-374和28位微型模块来说），看门狗定时器和恒定扫描定时器，定时器的三种功能块包括接通延时定时器，关段延时定时器，常开延时定时器（也叫时间定时器）。四种分时触点时间分别为0.01s,0.1s,1.0和1min

耗时时钟

在CPU电源打开的时候，耗时型时钟以100ms为单位来跟踪外部时间。一旦电源关闭的时候，这个时钟不能保持，一旦每秒硬件请求CPU中断使能的这一秒被记录，在时钟开始记时后，这一秒将震动大概100年。
系统软件工作和定时器功能块是耗时型时钟工作的基础，它不能通过用户程序或者用户复位，尽管这样，通过请求服务16，应用程序可以读出耗时型时钟现在的值。

日历时钟

:

在28位微型和90-30系列PLC331和更高模块中，一天的时间由日历时钟来保持，日历时钟保持了7个时间函数功能。

- 年(两位数字量)
- 月
- 每月中的一天
- 小时
- 分
- 秒
- 每周中的一天

日历型时钟有一个备用电池，当断电时，它能够依靠这块电池保持当前值，尽管这样，除非你初始化它，否则它的值是无效的。使用SVCREQ#7，应用程序可以读和设置日历时钟。

日历时钟能够能够通过CPU的重新配置软件和手动编程器中读取和设置。尽管这样，从CPU（350-364）的软硬件版本8.00开始，假如CPU显示随机存取存贮器的分配信息，保护参数设置为使能，假如CPU键锁选择开关选择在开的位置，手动编程器不能改变TOD时钟。应用CPU350-374时候，注意键锁选择开关的特征。（其他型号的CPU没有键锁选择开关）
键锁选择开关设置了手动月-月和年-年转换，它将自动补偿跳过的时间直到2079年。

看门狗定时器

在90-30系列PLC中，看门狗定时器用于导致扫描时间过长的严重故障，对CPU311-341，看门狗定时器的时间值为200毫秒，对CPU 350-374，看门狗定时器的时间值为500毫秒。这个是不能改变的固定值。在每次扫描开始的时候，看门狗定时器总是从0开始。

对90-30系列中331和更低版本的CPU来说，假如看门狗定时器的值溢出，OK LED将会熄灭，CPU将会被复位和关闭，输出将会是他们的默认值。任何形式的通讯都将不可能，所有微处理器都将停止，为了恢复，必须重新上电包括CPU。在90-20，系列90Micro和340以及更高版本的90-30系列CPU中，看门狗溢出使CPU复位，执行上电自检程序。当看门狗故障时，变成STOP模式。

断电定时器：

断电定时器用于确定PLC断电时间，当PLC断电的时候，它复位变成0并且开始记时。当PLC上电的时候，计时停止并且保持这个值，在第12章中的SVCREQ＃29能被用来读取这个定时器的值。

注意

当90-30系列PLC工作在常规扫描模式的时候，常规扫描定时器控制程序扫描的长度，在这种操作模式下，每次扫描需要用相同的时间，常规扫描定时器的值可以通过编程序人员来设定，而且能够是从5到看门狗定时器的值之间的任何一个值。常规扫描时间的默认值为100毫秒，作为特色地，对大部分应用程序来说，每次扫描，输入扫描，应用程序逻辑扫描，输出扫描都不需要确定相同的执行时间。

假如在没有完成扫描和预扫描没有跳过扫描时，常规扫描定时器停止，这时候在PLC错误表中，PLC会发出一个跳过扫描报警。在下次扫描开始的时候，PLC设置OV-SWP错误的连接，当PLC不在常规扫描定时模式的时候或者上次扫描的时间没有超过没有达到常规扫描定时器时，OV-SWP连接将被复位。

分时触点：

90系列的PLC提供了时间在0.01s,0.1s,1s,1min之间的四种分时触点。当PLC扫描开始系统初始化时这些分时触点的状态值可以改变。这些分时触点提供一个相同时间ON和OFF脉冲。这些分时触点是T_10MS (0.01秒), T_100MS (0.1秒), T_SEC (1.0秒), 和 T_MIN (1分).

分时触点时序图。

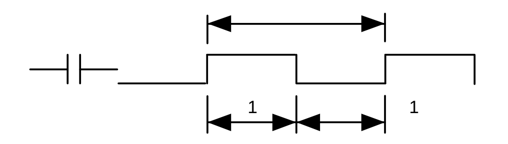


图2-6. 分时触点的时序图

第五节：系统安全

90-30系列，90-20系列和卫星PLC的安全设计用来阻止未经授权而修改PLC的内容。在PLC中有四个层次的可利用安全方式。第一层次也是常用的一种就是仅仅让PLC的内容可以读而不能修改它，其他的三种方式都是通过设置密码的方式来达到他们所在层次的安全。

每一个优先权高的曾相对于优先权低的层来说，都有更大的可以改变的能力，优先级别在那累积以一个级别被允许的优先是那一个级别的一个组合, 加上所有的较低级别。 级别和他们的优先权是::

Privilege Level	Description
第一层	除了密码外的所有数据都有可能被读到。这包括所有的数据记录单元（%I, %Q, %AQ，%R等）.，错误表，所有的程序类型块（数据，值，常量）PLC中的值都不可以改变
第二层	这一层允许写入数据记录单元 (%I, %R,等.).
第三层	这一层允许仅仅在停止模式下写入应用程序..
第四层	对系统来说，这是没有密码设置的默认层。有密码设置的默认层是更高的，无保护的层，这个更高的层 允许在运行和停止模式下，读和写所有记忆单元和密码。（重新配置的数据单元在运行模式下不允许更改）

密码：

在PLC中，对于每一层都有一个密码设置（除了第一层之外），每一层的密码是单一的。尽管这样，对于不同层可以使用同一个密码，密码的长度为1-4个ASII码。他们只能通过程序软件或者是手持编程器来进入或者修改。

只要PLC和编程器连在一起，就可以更改优先权。不需要任何激活，但是通讯槽必须没有损坏。假如在15分钟内没有通讯，优先层将变成没有保护的最高层。

和PLC连接上后，编程软件向PLC请求每一个优先层的保密状态。然后编程软件请求PLC移动到没有保护的最高层，因此不必请求任何一个特殊层，编程软件可以到达保护的最高层。然后手持编程器和PLC连接上，PLC回复这个保护的最高层。

改变优先权的请求：

编程器通过提供一个新的优先权和密码来请求改变当前优先权。假如编程器输入的密码不符，那么这次优先权的更改是无效的。现有的优先权是可以保持而且是没有更改的。假如你没有适当的优先权而且又想通过手动编程器进入或者修改PLC中的信息，手动编程器将发出一个错误的信息，请求将失败。.

锁定/解锁子程序

子程序可以通过软件中程序块的可锁特征来锁定/解锁子程序。我们可以利用的有两种类型的锁存子程序。

锁的类型	描述
只读	一旦锁住，你不能进入子程序
编辑	一旦锁住，子程序中的信息不能被编辑

预先设定的只读锁存或是编辑锁存子程序将不被锁在块程序的决定编辑中，除非他们都永久的是。

在可视的锁存子程序中将有一个查询或是查询取代函数,假如寻找的目标在可视的锁存子程序中，下面这两句话中一句将出现取代逻辑程序：

```
Found in locked block <block_name>   (Continue/Quit)
```

或者

```
Cannot write to locked block <block_name>  (Continue/Quit)
```

你可以继续查询和放弃查询.

包含锁存子程序的文件夹可能被清除或者删掉，假如一个文件夹包含锁存的子文件
当软件被复制，做备份，或者复制文件夹的时候，这些块将保留

永久性锁定子程序

除了VIEW LOCK 和EDIT LOCK,还有两种永久的的锁存。假如一个PERMANENT VIEW LOCK被设置，所有单独调用子程序都是无效的。假如一个PERMANENT VIEW LOCK被设置，将不可能编辑这个子程序块。.

注意

永久的锁存与常规的VIEW LOCK 和 EDIT LOCK是有区别的，一旦被设置成永久的锁存，他将永远不能更改。

一旦一个 PERMANENT EDIT LOCK 被设置，它将仅仅可能被更改为 PERMANENT VIEW LOCK. A PERMANENT VIEW LOCK不能更改成任何其他的锁存形式。.

第六节 系列 90-30，90-20 及 Micro PLC 的 I/O 系统

该PLC I/O系统在90-30系列与所支持的设备之间提供了接口。90-30 I/O模块直接插入其基板的插槽内。CPU的型号决定了所支持的I/O模块的数量：:

- 型号为350—374的CPU支持79个I/O模块, 它们还支持8个栈，其中包括7个扩展站和1个远程站。
- 型号为331，340，341的CPU支持49个I/O模块和5个站，其中包括4个扩展栈和1个远程站。
- 型号为311，313（5个槽的基板）的CPU支持5个I/O模块，型号为323（10个槽的基板）的CPU支持10个I/O模块，他们不支持扩展站和远程站。

90-30 PLC 的 I/O 结构如下图所示:

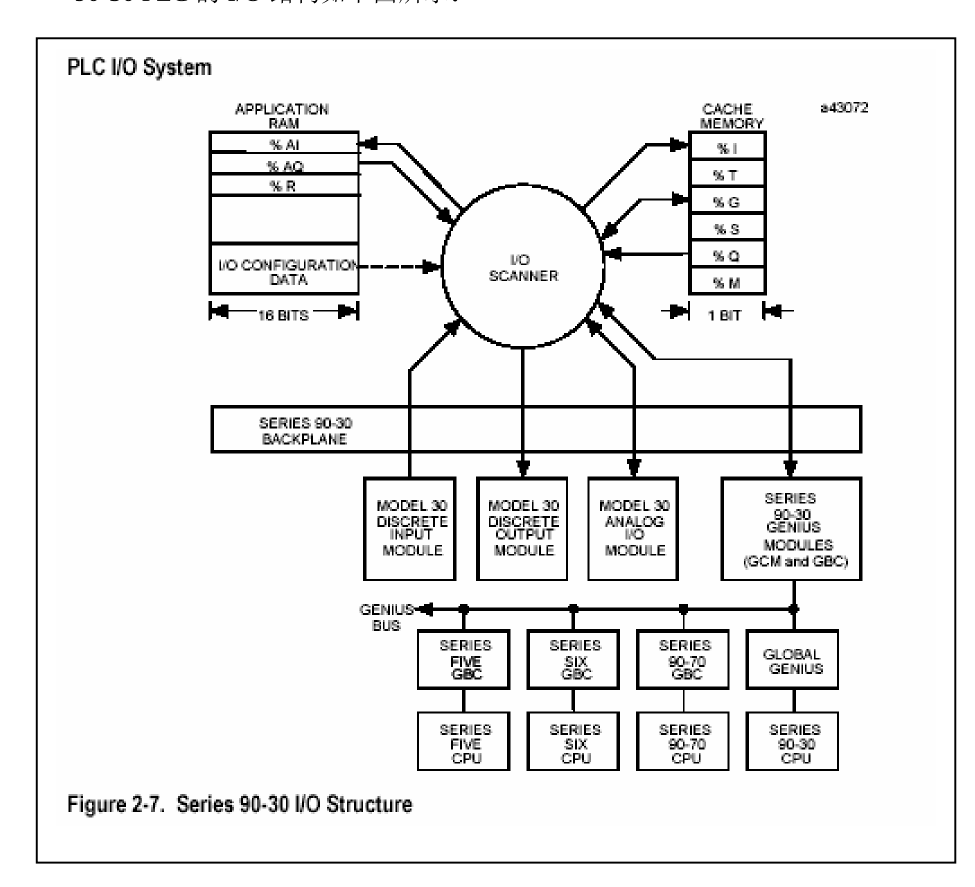


Figure 2-7. Series 90-30 I/O Structure

注意

上图为 90-30 I/O 的一种特殊结构。智能模块与选择模块不是 I/O 所扫描的部分；他们使用系统通讯窗口，想了解 90-20 I/O 结构的相关内容，请查阅 90-20 PLC 用户手册(GFK-0551).想了解 Micro PLC I/O 结构,请查阅 90Micro PLC 用户手册.

90-30 I/O模块

90-30 I/O模块有5种可用类型：数字输入,数字输出,模拟输入,模拟输出及选择模块.下表按照订货号、I/O点数、说明的不同将各个90-30 I/O模块列在了表中

注意

为了了解清单所列的模块的可用性，请与当地GE Fanuc的发行人联系。
查阅“Pub Number”可以了解每一个90-30 I/O模块的说明书与配线信息。

表2-8. 90-30 I/O模块

订货号	点	说明	出版号
		数字模块---输入	
IC693MDL230	8	120VAC独立的	GFK-0898
IC693MDL231	8	240 VAC独立的d	GFK-0898
IC693MDL240	16	120 VAC	GFK-0898
IC693MDL241	16	24 VAC/DC ±	GFK-0898
IC693MDL630	8	24VDC+	GFK-0898
IC693MDL632	8	125 VDC ±i	GFK-0898
IC693MDL633	8	24 VDC-	GFK-0898
IC693MDL634	8	24 VDC ±	GFK-0898
IC693MDL640	16	24VDC+	GFK-0898
IC693MDL641	16	24VDC-	GFK-0898
IC693MDL643	16	24VDC+,快速	GFK-0898
IC693MDL644	16	24VDC-,快速	GFK-0898
IC693MDL645	16	24 VDC±	GFK-0898
IC693MDL646	16	24 VDC ± 快速	GFK-0898
IC693MDL652	32	24 VDC ±	GFK-0898
IC693MDL653	32	24 VDC ±, 快速	GFK-0898
IC693MDL654	32	5/12 VDC±（TTL）	GFK-0898
IC693MDL655	32	24 VDC±	GFK-0898
IC693ACC300	8/16	输入模拟器	GFK-0898

表2-8. 90-30 I/O模块 – 接上页

订货号	点	说明	出版号
		Discrete Modules - Output	
IC693MDL310	12	120VAC ,0.5A	GFK-0898
IC693MDL330	8	120/240 VAC, 2A	GFK-0898
IC693MDL340	16	120 VAC, 0.5A	GFK-0898
IC693MDL390	5	120/240 VAC独立的，2A	GFK-0898
IC693MDL730	8	12/24VDC+,2A	GFK-0898
IC693MDL731	8	12/24VDC-,2A	GFK-0898
IC693MDL732	8	12/24VDC+,0.5A	GFK-0898
IC693MDL733	8	12/24VDC-,0.5A	GFK-0898
IC693MDL734	6	125VDC±,2A	GFK-0898
IC693MDL740	16	12/24VDC+,0.5A	GFK-0898
IC693MDL741	16	12/24VDC-,0.5A	GFK-0898
IC693MDL742	16	12/24VDC+,1A	GFK-0898
IC693MDL750	32	12/24VDC-	GFK-0898
IC693MDL751	32	12/24VDC+,0.3A	GFK-0898
IC693MDL752	32	5/24VDC(TTL)-,0.5A	GFK-0898
IC693MDL753	32	12/24VDC+/-,0.5A	GFK-0898
IC693MDL760	16	11充气的5个24VDC+,0.5A	GFK-1881
IC693MDL930	8	继电器，常开., 4A，隔离	GFK-0898
IC693MDL931	8	继电器, BC，隔离	GFK-0898
IC693MDL940	16	继电器，常开., 4A	GFK-0898
		输入输出模块	
IC693MDR390	8/8	24VDC输入，继电器输出	GFK-0898
IC693MAR590	8/8	120VDC输入，继电器输出	GFK-0898
		模拟模块	
IC693ALG220	4 ch	模拟量输入，电压型	GFK-0898
IC693ALG221	4 ch	模拟量输入，电流型	GFK-0898
IC693ALG222	16	模拟量输入，电压型	GFK-0898
IC693ALG223	16	模拟量输入，电流型	GFK-0898
IC693ALG390	2 ch	模拟量输出，电压型	GFK-0898
IC693ALG391	2 ch	模拟量输出，电流型	GFK-0898
IC693ALG392	8 ch	模拟量输出，电流型/电压型	GFK-0898
IC693ALG442	4/2	模拟量，电流/电压型输入输出	GFK-0898

表2-8. 90-30 I/O模块 – 接上页

订货号	说明	出版号
	选择模块	
IC693APU300	高速计数器	GFK-0293
IC693APU301 IC693APU301	轴APM模块,1-轴-从模式 轴APM模块,1-轴-标准模式	GFK-0781 GFK-0840
IC693APU302 IC693APU302 IC693MCS001/002*	轴APM模块,2-轴-从模式 , 轴APM模块,2-轴-标准模式 轴控制系统（1和2轴）	GFK-0781 GFK-0840 GFK-1256
IC693DSM302 IC693DSM314	轴数字伺服模块 轴数字伺服模块	GFK-1464 GFK-1742
IC693APU305 IC693CMM321	I/O处理器模块 以太通讯模块	GFK-1028 GFK-1541
IC693ADC311	数字\文字显示协处理器	GFK-0521
IC693BEM331	Genius总线控制器	GFK-1034
IC693BEM320	I/O Link 接口模块(从)	GFK-0631
IC693BEM321	I/O Link 接口模块(主))	GFK-0823
IC693CMM311	通讯协处理器模块	GFK-0582
IC693CMM301	Genius通讯模块	GFK-0412
IC693CMM302	增强型Genius通讯模块	GFK-0695
IC693PBM200	Profibus 主模块	GFK-2121
IC693PBS201	Profibus 从模块	GFK-2193
IC693PCM300	PCM,160K Bytes(35K 用户MegaBasic 程序)	GFK-0255
IC693PCM301	PCM,192Bytes(47k用户MegaBasic 程序)	GFK-0255
IC693PCM311	PCM,640Bytes(190k 用户MegaBasic 程序)	GFK-0255
IC693PTM100/101	变频器电源模块)	GFK-1734
IC693TCM302/303	温度控制模块(TCM),8 信道	GFK-1466

* Obsolete. Listed for reference only.

I/O 数据格式

数字输入输出以位的形式存储在高速缓存里。模拟输入输出数据以字的形式存储，其内容驻留在一部分应用 RAM 中便于分配。

90-30 输出模块缺省条件

在上电时，90-30数字输出模块默认为无输出。该默认值将保持直到PLC第一次输出扫描。模拟输出模块可以通过端子上的跳线进行组态，其值可以是0，也可以保持上一个状态值。所以，模拟输出模块也可以通过外部电源供电，这样，即使PLC掉电，模拟输出模块也可以继续工作在已选定的默认值下。

数据诊断

诊断位可以用在%S的内存里，他将显示I/O模块的错误或者一个I/O组态的失配。诊断信息不能用在各自的I/O点。关于默认值设置的更多信息可参阅第3章，“默认值的解释与修正”

全局数据

Genius全局数据

IC693CMM301,受限于固定的%G地址，并且从SBA16—32的每个总线地址中只能交换32位。当再次安装时，我们不推荐使用这款模块；而是推荐使用增强型的GCM，它具备更好的性能。当多CPU调用Genius全局数据时，90-30 PLC可支持数据的快速共享。在CPU中，第五版或更新的Genius总线控制器IC693BEM331,和增强型的Genius通讯模块IC693CMM302,能够将128位的数据发送到其他PLC或是电脑中。他们可以在网络上从超过30台的Genius控制器那里接收128位的数据。数据可以任何存储形势被发送或接收。最初的Genius通讯模块

以太网通讯

T364型CPU(9.0或更新版)可以通过两个网口的任意一个建立以太网的连接。并且提供AAUI和10BaseT网口。374型CPU通过两个10BaseT/100BaseTx自动连通全双工以太网口与网络进行连接.

364型和374型CPU都支持以太网全局数据（EGD）,就像一台传送数据的设备被允许传送数据给一台或多台设备。Logicmaster 90 不支持EGD。(它需要一个基于Windows的程序提供给90 PLC).

系列 90-20 I/O 模块

以下 I/O 模块可用于 90-20 PLC.每个模块按照订货号、I/O 点数、说明的不同列在表格中。I/O 和电源一起被整合到了基板上。要了解各模块的说明及配线情况，可查阅《90-20 可编程控制器用户手册》第 5 章，GKF-0551.

订货号	详细描述	I/O点
IC692MAA541	I/O及电源基板模块 120VAC进/120VAC出/120VAC电源	16输入/12输出
IC692MDR541	I/O及电源基板模块 24VDC进/继电器输出/120VAC电源	16输入/12输出
IC692MDR741	I/O及电源基板模块 24VDC进/继电器输出/120VAC电源	16输入/12输出
IC692CPU211	CPU模块，型号CPU211	不可用

组态及编程

组态就是分配逻辑地址和特性给系统内部的硬件模块的过程。可以使用组态软件或手持型编程器，在程序编写前或编写后进行组态。不过推荐在编程前完成。查阅用户手册可以了解编程软件的详细内容，关于如何创建、传送、编辑及打印程序。第 4 章通过对 12 个编程命令的描述，可用来为 90-30 和 90-20 可编程控制器创建一系列逻辑程序。

.

Chapter 3 第三章　故障说明和校正

本章描述内容有助于发现并校正系列90-30, 90-20, 和 Micro PLC系统故障。本章包括PLC故障表中的故障说明和I/O故障表中的故障类别。

每一故障的解释都列出了它在PLC故障表中的故障说明或是在I/O故障表中的故障类别。从而找出与你的编程器中显示相对应的故障说明或故障类别。其中介绍了引起故障的原因以及解决故障的指导说明。

第三章 包含以下各个部分：

第一节 故障处理

注意

本章描述的故障处理信息适用于使用Logicmaster 90-30/20/Micro编程软件编程的系统。

当影响系统工作和运行的故障或状况出现时，系列90-30, 90-20, 或系列90 Micro PLC 系统将会报错。这些状况，包括诸如I/O模块或机架损坏等可能影响到PLC控制机器或过程的状况。 或者，这些状况仅仅作为报警，例如指示备份电池需要更换的电池电压低信号。然而，一些故障表中报告的状况并非故障。例如PLC系统中增加了一个新模块，I/O故障表中将会列出“增加I/O模块”。

报警处理器

故障就是故障或状况本身。当CPU接收并且处理的故障，被称作**报警**。CPU中处理这些情况的固件被称作报警处理器。报警处理器用户接口通过编程软件。在实际应用中，任何记录到记录到故障表中的故障会在PLC故障中或I/O故障表中显示。

故障分类

系列90-30, 90-20, 和Micro PLCs 检测以下几类故障。这些包括内部故障，外部故障和运行故障。

故障类别	示例
内部故障	模块不响应 电池电压低信号 存储器校验和错误
外部 I/O 故障	机架或模块损坏 增加模块或机架
运行故障	通讯故障 配置故障 密码错误

注意

有关Micro PLC故障处理更多信息, 参见系列*90 Micro PLC 用户手册*(GFK-1065).

系统故障响应

硬件故障要求不是系统关机，就是系统可容。I/O故障为PLC所允许,但是可能它们不被应用程序或控制过程所允许。运行故障一般被允许。系列90-30, 90-20, 和 Micro PLC 故障有以下两种特征：

特征	说明
故障表	I/O 故障表 PLC 故障表
故障作用	致命性 诊断性 提示性

故障表

PLC中提供两个故障表用来记录故障，I/O故障表记录I/O有关故障，PLC故障表记录所有其它故障。下表列出故障分组，故障作用，受影响的故障表，及系统变量%S的别名。

Table 3-1. 故障一览表

故障分组	故障作用	故障表	专用离散故障变量			
I/O Module丢失或损坏	诊断性	I/O	io_flt	any_flt	io_pres	los_iom
可选模块损失或丢失	诊断性	PLC	sy_flt	any_flt	sy_pres	los_sio
系统配置不匹配	致命性	PLC	sy_flt	any_flt	sy_pres	cfg_mm
PLC CPU 硬件故障	致命性	PLC	sy_flt	any_flt	sy_pres	hrd_cpu
程序校验和故障	致命性	PLC	sy_flt	any_flt	sy_pres	pb_sum
电池电压低	诊断性	PLC	sy_flt	any_flt	sy_pres	low_bat
PLC 故障表填满	诊断性	—	sy_full			
I/O 故障表填满	诊断性	—	io_full			
应用程序故障	诊断性	PLC	sy_flt	any_flt	sy_pres	apl_flt
没有用户程序	提示性	PLC	sy_flt	any_flt	sy_pres	no_prog
用户RAM损坏	致命性	PLC	sy_flt	any_flt	sy_pres	bad_ram
口令访问错误	诊断性	PLC	sy_flt	any_flt	sy_pres	bad_pwd
PLC软件故障	诊断性	PLC	sy_flt	any_flt	sy_pres	sft_cpu
PLC存储故障	致命性	PLC	sy_flt	any_flt	sy_pres	stor_er
恒定扫描时间超时	诊断性	PLC	sy_flt	any_flt	sy_pres	ov_swp
未知的PLC故障	诊断性	PLC	sy_flt	any_flt	sy_pres	
未知的I/O故障	诊断性	I/O	io_flt	any_flt	io_pres	

故障作用

故障可以是致命的，诊断的，或者提示性的。

致命性故障被记录到相应的故障表中，诊断变量将被置位，系统被中止。诊断故障被记录到相应的故障表中，诊断变量被置位。提示性故障仅仅记录到相应的故障表中。

下表列出故障作用：

Table 3-2. 故障作用

故障作用	由CPU响应
致命性	记录到故障表中 设置故障变量 转入**停止**模式
诊断性	记录到故障表中 设置故障变量
提示性l	记录到故障表中

当检测到故障时，CPU 执行上表中相应动作。故障作用在系列90-30 PLC, Series 90-20, 或系列 90 Micro PLC不可配置.

故障参考地址

系列90-30 的故障参考地址属于一类 – 专用故障变量。 故障参考地址用来表示故障的发生。故障参考地址保持ON，直到PLC内存被清除或故障在应用程序中被清除。

以下是故障位被置位后被清除的例子。本例中，当表示系统扫描超时的系统变量OV_SWP(%SA0002)接通时，线圈Light_01为ON。如果%I0359接通，OV_SWP和线圈Light_01将为OFF, 因为%I0359接通后将OV_SWP复位。

```
| ov_swp                                                              Light_01
|——] [——————————————————————————————————————————————————————————————————( )—
|
|%I0359                                                               ov_swp
|——] [——————————————————————————————————————————————————————————————————(R)—
|
```

系统状态变量

报警处理器中包括128个S%系统状态变量。这些状态变量指出故障出处和故障类型。状态变量存在%S, %SA, %SB, 和 %SC 中, 每个变量有一个别名。例如，%SA0009 别名是CFG_MM,它接通表示配置不等。根据需要可以在应用程序中使用这些变量，参考第二章“系统运行”中系统状态列表。

其它故障影响

本章描述的两类故障还有一些其它的故障影响。下表中将讨论这些影响：

故障	说明
PLC CPU 软件故障	如果PLC CPU故障被记录，系列90-30 或 90-20 CPU 立即转入 **ERROR SWEEP**模式。不允许任何操作。解决该故障唯一办法重新上电复位PLC。
PLC 顺序存储故障	如果在向PLC存储过程中，PLC和编程器的通讯被中断或出现其它任何中止下装（存储）的故障时，记录PLC Sequence Store Failure故障。只要该故障仍然存在，PLC将无法转入**RUN**模式。为了恢复运行应清除故障，可以通过编程器清除故障表。

PLC 故障表显示

PLC故障表显示诸如口令错误，PLC配置不匹配，奇偶校验错误，通讯故障等故障。

如果编程软件为**OFFLINE** 模式，故障存储在PLC中，故障表中没有故障显示。如果编程软件为**ONLINE** 或 **MONITOR**模式，故障表中显示故障数据。如果是**ONLINE**模式，可以清除故障，可以配置口令保护。

一旦故障被清除后，仍然存在的故障将不再记录到故障表中 (除了“电池电压低”故障)除非PLC重新上电或下装了新的配置。

I/O 故障表显示

I/O 故障表显示诸如线路故障，地址冲突，强制回路，和 I/O总线故障等故障.

如果编程软件为**OFFLINE** 模式，故障存储在PLC中，故障表中没有故障显示。如果编程软件为**ONLINE** 或 **MONITOR**模式，故障表中显示故障数据。如果是**ONLINE**模式，可以清除故障，可以配置口令保护。

一旦故障被清除后，仍然存在的故障将不再记录到故障表中，除非PLC重新上电或下装了新的配置。

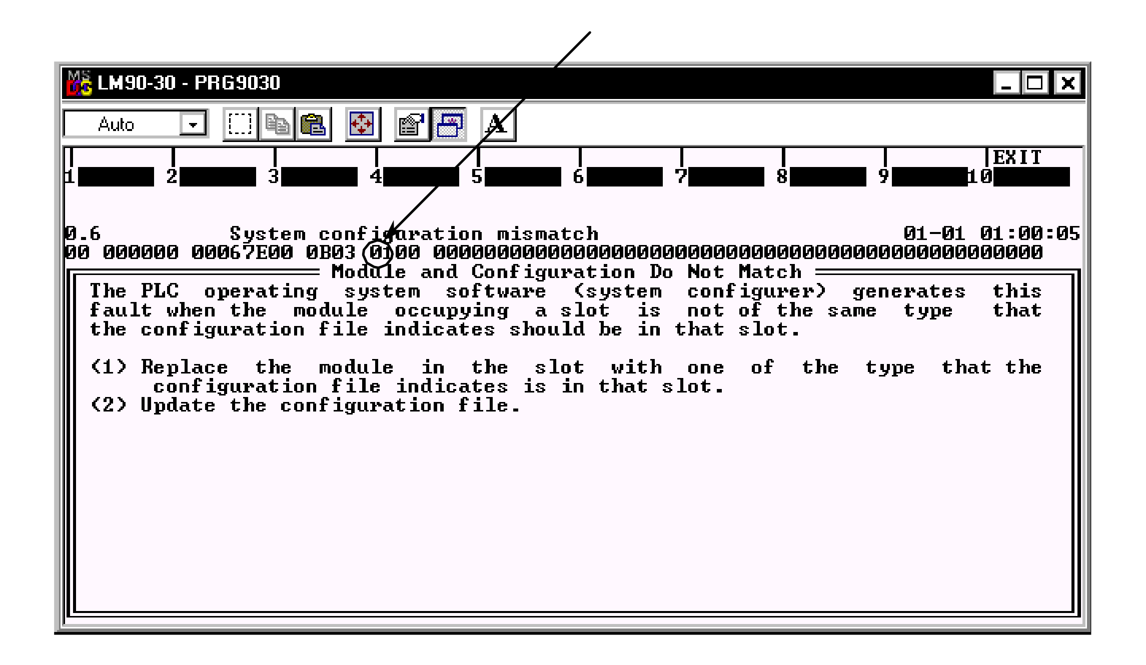
访问其它故障信息

故障表包含与故障有关的基本信息。每条故障的其它信息可以通过编程软件显示。另外编程软件为每条故障提供一个十六进制错误代码。

本章每一故障解释最后一项“校正”，列出了解决故障的方法。注意该解决方法有些故障包括以下语句：**Display the PLC Fault Table on the Programmer. Contact GE Fanuc Field Service, giving them all the information contained in the fault entry。** 第二句意味着你必须告知现场服务商可从故障表中直接读到的信息和十六进制错误代码。现场服务人员将给你进一步的解决办法。

下图包含其它故障信息和十六进制故障代码。(故障代码是左起第五组数的前两个十六进制数) 为了进入该画面，将光标移到该故障上(I/O模块损坏) 然后按下F10即可进入“zoom” 。返回故障表画面，按下Escape key或Shift＋F10即可。

Hexadecimal Fault Code



第二节　PLC故障表解释

每个故障解释包括故障说明和解决故障办法。一些故障说明有多个原因，错误代码还显示其它故障信息，这些信息用于区别同一故障说明的不同故障。下例中错误代码是这组数字中第五组（从左边开始）的前两个十六进制数。

```
01   000000   01030100   0902   0200   000000000000
                                |
                                |_____  Error Code  (first two hex
                                           digits in fifth group)
```

某些故障可能由于访问PLC CPU内存故障引起。这些故障可能由系统断电和电池电压低等引起。为了避免重复由内存损坏引起的故障解决办法简单描述如下：

Perform the corrections for Corrupted Memory.

该表述含义:

1. 若系统断电，更换电池。电池电压低不足以保存存储器中的内容。
2. 更换PLC CPU。PLC CPU板上的集成电路可能已损坏。

下表能帮助你尽快找到PLC故障解释。每一项按在编程器中的显示列出来。

故障说明	页码
丢失或损坏，可选模块	3-8
重新安排，增加，或多余的，可选模块	3-8
系统配置不匹配	3-9
可选模块软件故障	3-10
程序块校验和故障	3-10
电池电压低信号	3-10
恒定扫描时间超时	3-11
应用程序故障	3-11
无应用程序	3-12
上电时应用程序损坏	3-12
口令错误	3-12
PLC CPU 系统软件故障	3-13
存储过程中通讯故障	3-15

故障作用

- **致命性** 故障发生时在发生该故障的这个扫描周期结束时使PLC转入**STOP**模式。
- **诊断性** 故障记录到故障表中，相应的故障变量被置位; PLC仍然为**RUN**模式。
- **提示性** 故障仅仅记录到故障表中; PLC仍然为**RUN** 模式。

丢失或损坏，可选模块

当可选模块没有相应时，报故障**Loss of, or Missing Option Module**。该故障发生在上电时模块丢失或运行过程中模块没有响应。该故障属于**诊断性**故障。

故障代码:	1, 42
名称:	可选模块软件复位失败
说明:	软复位后（按下复位按钮）PLC CPU 不能与可选模块建立通讯.
校正:	(1) 重复软复位。 (2) 更换可选模块 (3) 系统断电。确认模块在机架中接触良好，并且所有电缆连接正确，接触良好。 (4) 检查或更换电缆。
故障代码:	所有其它
名称:	组态过程中模块故障
说明:	系统上电或存储配置时模块故障
校正:	(1) 系统断电。更换模块。

复位，增加的，额外的，可选模块

当可选模块(PCM, ADC, etc.) 在线被复位，或机架中模块在配置中没有配置时,产生错误**Reset of, Addition of, or Extra Option Module** 。该故障是**诊断性**故障。

校正:	(1) 更新配置文件，将该模块添加至配置中。 (2) 从系统中拆去该模块.

系统配置不匹配

当模块与配置文件中对应槽位的模块不相符时，产生故障**Configuration Mismatch** 。该故障是**致命性**故障。

错误代码:	1
名称:	系统配置不匹配
说明:	当模块与配置文件中对应槽位的模块不相符，或配置机架类型与实际机架类型不相符时，产生此故障。
校正:	确认不相符原因重新配置
错误代码:	6
名称:	系统配置不匹配
说明:	同1 当模块与配置文件中对应槽位的模块不相符，或配置机架类型与实际机架类型不相符时，产生此故障。
校正:	确认不相符原因重新配置
错误代码:	18
名称:	硬件不支持
说明:	PCM 或PCM型模块置于CPU 311, 313, 或 323 系统中，或在扩展机架或远程机架中。
校正:	取下PCM 或PCM型模块或安装支持该模块的CPU 。 注意：该模块必须置于CPU所在机架中，且该CPU必须支持该模块。
错误代码:	26
名称:	模块忙–配置未被模块接受
说明:	模块此时不能接受配置因为模块正忙于其它进程。
校正:	允许模块完成当前操作，重新存储配置。
错误代码:	51
名称:	执行END功能，该END功能来自SFC
说明:	在SFC或逻辑中END功能被SFC调用产生此故障。
校正:	从SFC中或逻辑中删除该END功能。

可选模块软件故障

当PCM或ADC模块发生不可恢复软件故障时产生此故障**Option Module Software Failure**。该故障是**致命性**故障**。**

错误代码:	全部
名称:	COMMREQ频率太高
说明:	COMMREQs 发送至模块速度太快.
校正:	修改PLC程序，以较低速率向模块发送

程序块校验和故障

等PLC CPU检测到程序块（由编程软件下装）中的错误状态产生此故障**Program Block Checksum Failure**。也可能是系统上电存储器检查时或**RUN**模式时后台检测时PLC CPU检测到校验和错误时产生。该故障是**致命性**故障。

错误代码:	所有
名称:	程序块校验和故障
说明:	程序块损坏时，产生此故障。
校正:	(1) 清PLC内存，重新储存。 (2) 在故障表中显示故障，和GE的PLC现场服务商联系，告知故障表中所有记录信息。

电池电压低信号

当PLC CPU检测到电池电压低或模块如PCM等报告电池低时，产生此故障**Low Battery Signal** 。该故障是**诊断性**故障。

错误代码:	0
名称:	电池故障信号
说明:	CPU模块(或其它配有电池的模块)电池耗尽。
校正:	更换电池，系统无须断电。
错误代码:	1
名称:	电池电压低信号
说明:	CPU或其它模块电池电压低信号。
校正:	更换电池，系统无须断电。

恒定扫描时间超时

PLC CPU为**CONSTANT SWEEP**模式且扫描时间超过指定扫描时间时，产生此故障**Constant Sweep Time** 。故障附加数据包括头两个字节表示的实际扫描时间和后八个字节表示的程序名称。该故障是**诊断性**故障。

校正:	(1)	延长恒定扫描时间
	(2)	从应用程序中删除部分程序

应用程序故障

当PLC CPU检测到用户程序中故障时，产生此故障**Application Fault** 。该故障是**诊断性**故障。调用子程序嵌套溢出，此时是**致命性**故障。

错误代码:	7
名称:	调用子程序嵌套溢出
说明:	子程序调用最多8级，子程序调用子程序，依次调用最多不超过8级。
校正:	修改程序，使得子程序调用不超过8级。
错误代码:	1B
名称:	由于PLC内存限制，通讯请求不被处理。
说明:	非等待通讯请求被设置在队列中快于它们被处理(如每次扫描处理一次).这种情况下，当通讯请求建立达到PLC小于有效存储器的最小值这一点时，通讯请求将被中止不执行。
校正:	减少通讯请求或减轻和系统交换数据量
错误代码:	5A
名称:	用户关机请求
说明:	当应用程序执行SVCREQ #13(用户关机) 功能时，产生此故障。
校正:	无须处理。仅作为信息警报。

无用户程序

当PLC CPU从**STOP**转入**RUN**模式，或向PLC存储且PLC中无用户程序时，产生该故障**No User Program Present** 。当重新上电时PLC CPU 检测缺少用户程序。该故障是**提示性**故障。

校正:	进入**RUN**模式前，下载应用程序。

上电时用户程序受损

当PLC CPU检测到损坏的用户RAM时，产生该故障**Corrupted User Program on Power-Up。**PLC CPU维持**STOP**模式，直至下载有效的应用程序和配置文件。该故障是**致命性**故障。

错误代码:	1
名称:	Corrupted User RAM on Power-Up
说明:	The PLC operating software (operating software) generates this error when it detects corrupted user RAM on power-up.
校正:	(1)重新下装配置文件，用户程序，变量表(如果有) (2)更换PLC CPU电池 (3)更换PLC CPU扩展内存板 (4)更换PLC CPU
错误代码:	2
名称:	检测到非法布尔操作码
说明:	当在用户程序中检测到一错误指令时，PLC产生该故障
校正:	(1)重新下装用户程序和变量表(如果有)。 (2)更换PLC CPU扩展内存板 (3)更换PLC CPU

口令访问失败

当PLC CPU接收到请求改变一新优先级且口令对此优先级无效时，产生该故障**Password Access Failure**。该故障是**提示性**故障。

校正:	使用正确的口令重新请求。

PLC CPU 系统软件故障

故障组**PLC CPU System Software Failure** 由系列90-30, 90-20 或Micro PLC CPU固件产生。该故障可能在程序运行中多次出现。当出现该**致命性**故障时, PLC CPU**立即**转入**ERRORSWEEP**模式。当PLC在此模式下，不允许其它操作。解决这种故障办法：PLC重新上电。该故障是**致命性**的。

错误代码:	1 到 B
名称:	用户内存不能被分配
说明:	PLC(内存管理器)产生此故障，当请求从非法RAM中分配或不再分配内存给程序块时。在正式发布产品中不会出现此故障；该故障通常在制造商处进行固件升级时发生。
校正:	故障表中显示故障信息。和GE现场服务商联系，告知故障表中所有记录信息。

错误代码:	D
名称:	系统内存无效
说明:	PLC (I/O扫描器)产生此故障，程序块请求系统内存被拒绝，因为没有可用的系统内存。如果执行DO I/O功能块时发生该故障，那么它是*提示性*的。如果在上电初始化或自动配置时发生该故障，那么它是*致命性*的。
校正:	故障表中显示故障信息。和GE现场服务商联系，告知故障表中所有记录信息。

错误代码:	E
名称:	系统内存不能释放
说明:	PLC (I/O扫描器) 产生此故障，当请求内存管理器不在分配内存给程序块时且该请求失败。该故障仅在执行DO I/O 功能块时产生。
校正:	(1) 故障表中显示故障信息。和GE现场服务商联系，告知故障表中所有记录信息 (2) 修复损坏内存。

错误代码:	10
名称:	无效的I/O扫描请求
说明:	PLC (I/O 扫描器) 产生此故障，当系统或DO I/O功能模块的扫描请求，既不是全部I/O扫描，也不是部分I/O扫描。这在产品系统中不应发生。
校正:	故障表中显示故障信息。和GE现场服务商联系，告知故障表中所有记录信息。

错误代码:	13
名称:	PLC运行软件故障
说明:	PLC运行软件产生该故障，当某PLC运行软件出现问题时。这种故障不应在正式发布产品中出现；它们通常在制造商进行固件升级时出现。
校正:	(1)故障表中显示故障信息。和GE现场服务商联系，告知故障表中所有记录信息。 (2)修复损坏内存。

错误代码:	14, 27
名称:	PLC程序存储器损坏
说明:	PLC运行软件产生该故障，当某PLC运行软件出现问题时。这种故障不应在正式发布产品中出现；它们通常在制造商进行固件升级时出现。
校正:	(1)故障表中显示故障信息。和GE现场服务商联系，告知故障表中所有记录信息。 (2) 修复损坏内存。

错误代码:	27 到 4E
名称:	PLC运行软件故障
说明:	PLC运行软件产生该故障，当某PLC运行软件出现问题时。这种故障不应在正式发布产品中出现；它们通常在制造商进行固件升级时出现。
校正:	故障表中显示故障信息。和GE现场服务商联系，告知故障表中所有记录信息。

错误代码:	4F
名称:	通讯失败
说明:	PLC (服务请求处理器)产生该故障，当它试图响应需要背板通讯的请求，但该请求被拒绝时。
校正:	(1)检查总线有无非正常现象。 (2)更换发出请求的智能可选模块。

错误代码:	50, 51, 53
名称:	系统内存故障
说明:	PLC产生该故障，当程序块请求系统内存被拒绝，因为没有可用的系统内存或内存出错。
校正:	(1)故障表中显示故障信息。和GE现场服务商联系，并告知故障表中所有记录信息。 (2)修复损坏内存。

错误代码:	52
名称:	背板通讯失败
说明:	PLC (服务请求处理器)产生该故障，当它试图响应需要背板通讯的请求，但该请求被拒绝时。
校正:	(1) 检查总线有无非正常现象。 (2) 更换发出请求的智能可选模块。 (3) 检查编程电缆是否连接正确。

错误代码:	所有其它
名称:	PLC CPU 内部系统故障
说明:	系统内部故障一般不应出现在正式发布产品系统中。
校正:	故障表中显示故障信息。和GE现场服务商联系，并告知故障表中所有记录信息。

存储过程通讯故障

向PLC存储程序块和其它数据时，可能会产生此故障**Communications Failure During Store**。如果执行存储任务的编程设备通讯被终端或其它任何导致存储中断的故障，该故障被记录。只要该错误存在，控制器将不能转入**RUN**模式。

重新上电时该错误不会自动清除;用户必须指定专门的清除命令。该故障是致命性故障。更多信息，参见本章之前“其它故障影响”部分内容。

校正:	清除故障，重新存储程序或配置文件。

第三节: I/O 故障解释

I/O 故障表报告以下三类有关故障信息:

- 故障种类.
- 故障类型.
- 故障说明.

下页描述的故障只有故障种类，没有故障类型和故障组。

每个故障说明包括故障描述和解决此故障的方法。一些故障描述由多个原因引起。因此，错误代码包括额外信息，按下CTRL-F可得到，这些额外信息用于区分同一故障说明的不同故障(更多CTRL-F信息，参见本手册Appendix B,“故障表说明,”)。故障种类是第五组数中第一个十六进制数，如下所示。

```
02  1F0100  00030101FF7F  0302  0200   84000000000003
                           |
                           |_____  Fault Category (first two hex
                                      digits in fifth group)
```

下表能帮助你尽快找到I/O故障解释。每一项按在编程器中的显示列出来。

I/O模块损坏

故障种类 **Loss of I/O Module** 适用于30型PLC离散与模拟I/O模块。此类故障没有故障类型或故障说明。该故障是**诊断性**故障。

说明:	当PLC检测到30型I/O不响应来自PLC CPU的指令时，或配置文件中某槽中配有 I/O模块，但实际该槽并无模块时，产生此故障。
校正:	(1) 更换模块 (2)修改配置文件 (3)故障表中显示故障信息。和GE区域服务商联系，并告知故障表中记录所有信息。

其它I/O模块

故障种类**Addition of I/O Module**适用于30型PLC离散与模拟I/O模块。此类故障没有故障类型或故障说明。该故障是**诊断性**故障。

说明:	当I/O模块对操作反应故障时，产生此类故障。
校正:	(1)如果该模块已经被拆除或被更换，或远程机架已经上电，无须采取措施。 (2)更新配置文件或拆除模块。
说明:	当检测到30型I/O模块存在于机架某槽中，但配置文件中该槽空着时，产生此类故障。
校正:	(1) 拆除可能在错位的插槽中的模块。 (2)如果需要，更新配置文件，包括添加的模块。

Chapter 4 第四章 继电器功能模块

本节将讲述梯形逻辑回路中触点、线圈和链路的应用.

触点

触点是用来监控参考地址的状态。触点是否有电流流通取决于被控参考地址的状态或现状，或取决于触点的类型，如果参考地址的状态为1，则地址为ON，若其状态为0，则为OFF.

表 4-1. 触点类型

<table>
<tr><th>触点类型</th><th>显示</th><th>触点向右流过电流</th></tr>
<tr><td>常开</td><td>—| |—</td><td>当参考地址为ON.</td></tr>
<tr><td>常闭</td><td>—|/|—</td><td>当参考地址为 OFF.</td></tr>
<tr><td>延续触点</td><td><+>———</td><td>如果先行延续线圈为ON.</td></tr>
</table>

线圈

线圈用来控制离散参考地址。必须用条件逻辑来控制对线圈的电流流向。线圈将直接产生作用；它们不向右通过电流。如果程序中的附加逻辑应作为线圈条件的一个结果来执行的话，对可使用的那个线圈或一个延续线圈或一个延续线圈/触点组合来说，应采用一内部参考地址

线圈总是处于逻辑线（行）的最右边，一个梯级可以包含多达8各线圈。

线圈的类型将根据所需程序作用的类型采用，当电源为循环加入，或PLC从STOP模式切换到RUN模式，则保持线圈的状态便被存贮，而非保持线圈的状态则置为零。

表 4-2.　线圈类型

线圈类型	显示	线圈电流	结果
常开	—()—	ON OFF	设置参考地址为ON 设置参考地址为OFF
反线圈	—(/)—	ON OFF	设置参考地址为OFF 设置参考地址为ON
保持线圈	—(M)—	ON OFF	设置参考地址为ON，保持 设置参考地址为OFF，保持
反保持线圈	—(/M)—	ON OFF	设置参考地址为OFF, 保持 设置参考地址为ON，保持
正跳变	—(ϒ⎤)—	OFFϒ ⎜ON	如果参考地址为OFF, 置为ON一个扫描周期.
负跳变	—(ϒ⎱)—	ONϒ⎦OFF	如果参考地址为OFF, 置为ON一个扫描周期.
置位	—(S)—	ON OFF	设定参考地址为ON 直到被—(R)—复位 不改变线圈状态
复位	—(R)—	ON OFF	设定参考地址为OFF 直到被—(S)—置位. 不改变线圈状态
保持置位	—(SM)—	ON OFF	设定参考地址为ON 直到被 —(RM)—, 复位，保持. 不改变线圈状态
保持复位	—(RM)—	ON OFF	设定参考地址为OFF 直到被 —(SM)—, 置位，保持 不改变线圈状态
延续线圈	——<+>	ON OFF	设定下一个延续触点为ON. 设定下一个延续触点为OFF.

常开 —| |—

如果相关的参考地址为ON（1），则常开触点相当于一个接通的电源的开关。

常闭 —|/|—

如果相关的参考地址为OFF（0），则常闭触点相当于一个在流通的开关。

示例

下例给出了一个由10个单元（元素）组成的回行，其中，每个单元都有一个别名：E1到E10。当参考点E1、E2、E5、E6和E9为ON，而参考点E3、E4、E7和E8为OFF，则线圈E10为ON。

```
|
|  E1       E2       E3       E4       E5       E6       E7       E8       E9      E10
|——| |——————| |——————|/|——————|/|——————| |——————| |——————|/|——————|/|——————| |——————( )—
|
```

线圈 —( )—

当线圈接受电能时，它将一个离散地址置为ON，它时非保持的。因此，它不能与系统存贮器（%SA, %SB, %SC ）或全局Eenius参考地址（%G ）共用。

示例

在下例中，当参考点E1为ON，而参考点E2为OFF时，线圈E3为ON..

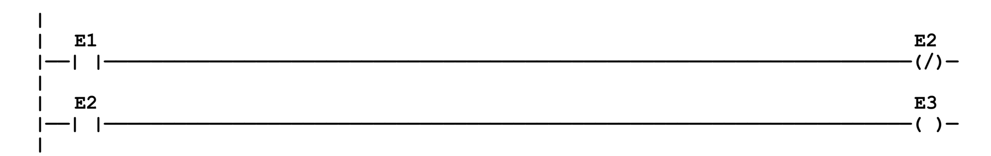
```
|
|  E1       E2                                                                           E3
|——| |——————|/|——————————————————————————————————————————————————————————————————————————( )—
|
```

求反线圈 —(/)—

求反线圈当其未接受电能时，则将一个离散地址置ON。它没有记忆性。因此它不能与系统存贮器（%SA, %SB, %SC ）或全局Eenius参考地址（%G ）共用。

示例

在下例中，当参考点E1为ON时，线圈E3为ON.



保持线圈 —(M)—

像一个常开线圈，当保持线圈接收电能时，便将一离散地址置为ON。电源掉电后，保持线圈状态被保存。因此，它不能同精确非保持存贮器（%T ）的地址共用。

非保持线圈 —(/M)—

当非保持线圈未接收电能时，它将一个离散地址置为ON。电源掉电后，非保持线圈的状态不被保持。因此，它不能同精确非保持存贮器（%T ）的地址共用。

正向跳变线圈 —(↑)—

如果与一个正向变换线圈相关的参考点是OFF，每当这个线圈同流变为ON，一直到下次线圈被执行为止（如果包含线圈的梯级则按时序扫描跳转，它仍将保持为ON）。这个线圈可用作一个单稳态触发器。

不要从外部设备（例如PCM、编程器、ADS等），写入正向变换线圈所用的参考地址，因为这将破坏这种线圈所具有的单稳定特性。

正向变换线圈可与来自不管是保持还是非保持型存贮器(%Q,%M, %T, %G, %SA, %SB, 或%SC)的地址共用。

负向跳变线圈 —(ϒ⌇)—

如果与该线圈相关的参考地址为OFF，且线圈停止接收电能，则参考地址置ON，直至下次线圈被执行为止。

不要从外部设备写入反向变换线圈所用的参考地址，因为这将破坏这种线圈所具有的单稳定特性。

反向变换线圈可与来自不管是保持还是非保持型存贮器(%Q,%M, %T, %G, %SA, %SB, 或%SC)的地址共用。

示例

在下例中，当参考地址E1由OFF转到ON，线圈E2和E3通流，则E2为ON，则有一次扫描。当E1由ON转到OFF，电流从E2和E3中消失，则E3为ON，则有一次扫描

```
|
|    E1                                                          E2
|——| |——————————————————————————————————————————————————————————(ϒ|)—
|
|     E1                                                         E3
|——| |——————————————————————————————————————————————————————————(ϒ⌇)—
|
```

置位线圈 —(S) —

置位和复位线圈使非保持型线圈，不能用来保持（锁定）一个参考地址（不论其为ON或为OFF）的状态。当一置位线圈通流，其地址将停留为ON（不论线圈自己是否持续通流），直到被其他线圈复位。

复位线圈 —(R)—

如果线圈通电，复位线圈将把一个离散参考地址置位OFF。地址将保持为OFF，直至由同一参考线圈置位。一对作后解决的置位先前或复位线圈具有优先权。

示例

在下例中，如果E2为ON，则E1被置位，为ON，不管E2转换为OFF，E1将保持为ON，直到被E3复位。

注意:　如果E2和E3在同一时刻都为ON使，线圈E1将被置为OFF。因为程序扫描从上到下，所以线圈的状态将在第二行中被复位，并被写到输出表中，如果行相反的话，在E2和E3同时为ON时，E1线圈将被置为ON。

```
|
|  E2                                                              E1
|--| |-------------------------------------------------------------(S)-
|
|
|
|
|  E3                                                              E1
|--| |-------------------------------------------------------------(R)-
|
|
|
|
```

注意

当线圈检验级别为SINGLE，你可使用仅带有一个线圈的一特定％M或％Q参考地址。但你能同时使用带有一置位线圈和一复位线圈。当线圈检验级别为WARN MULTIPLE或MULTIPLE,那么每个参考地址都能与复合线圈、置位线圈和复位线圈一起使用。带有多样用法的草考地址能被一置位线圈或一标准线圈转变至ON，并能被一复位先前或一标准线圈转变为OFF.

置位保持线圈　—(SM)—

保持置位线圈和复位线圈与置位和复位线圈类似。但虽然电源消失或当PLC由STOP转为RUN模式时，它们仍保持住。一个保持置位线圈当其通电时，将把一个离散地址置位ON,这个地址将一直保持为ON，直到由一保持复位线圈所复位为止。

保持置位线圈将为既定的参考地址，而把一个非定义结果写入转换位（参见第二章操作系统中关于“转换与超驰”的情况）。

复位保持线圈　—(RM)—

若这种线圈已通电，便将一个离散参考地址置为OFF。此地址将一直保持OFF，直到由一保持置位线圈所置位为止，虽然电源消失或当PLC由STOP转为RUN模式时，该线圈的状态均将保持住。

保持复位线圈将为既定的参考地址，而把一个非定义结果写入转换位（参见第二章操作系统中关于“转换与超驰”的情况）。

链回路

水平和垂直链路是用来连接功能模块间一条梯形逻辑中的各单元，其用途是为了沿一条逻辑线完成从左向右的逻辑（能量）流。

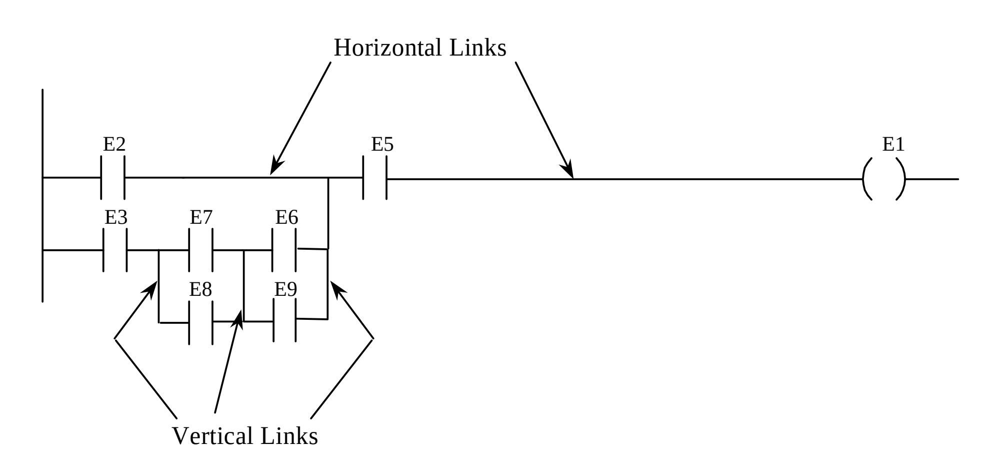
注 意

用户不得使用水平链路去直接连接左侧电源母线和功能块或线圈。不过用％S7，AWL_ON（总为ON）常开触点可连接电源母线来调用每次扫描得功能块。

示例

下例中用到的链路：

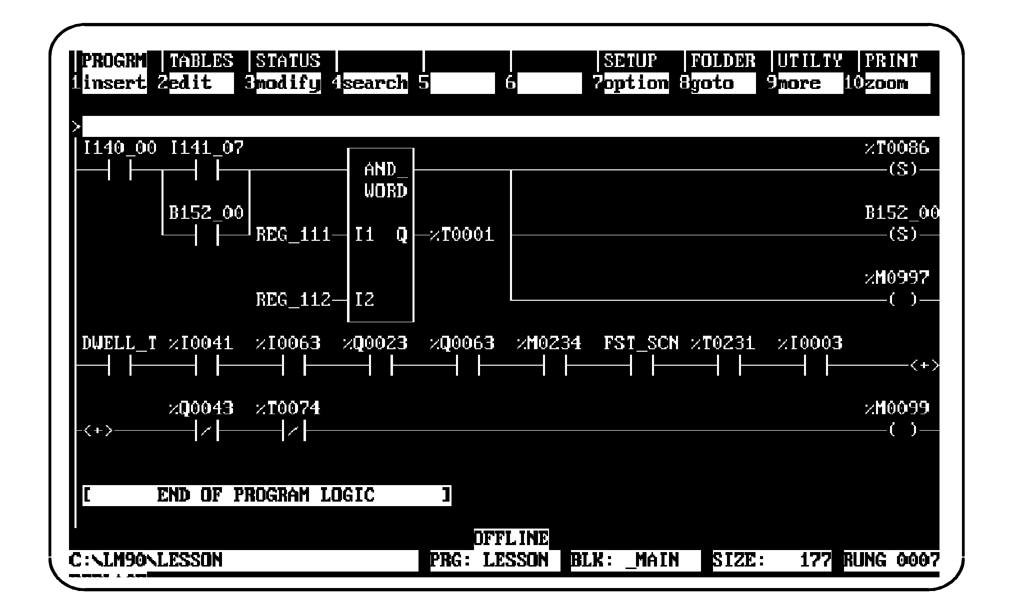
- 水平链路连接触点E2和E5，和E5到线圈E1。
- 垂直链路连接E8和E6，E9和E7，和右侧E7/E9到E2和E5的汇合处



延续线圈 (———<+>) 和触点 (<+>———)

延续线圈 (———<+>)和延续触点 (<+>———) 被用来将十列以内的继电器梯形图的回路逻辑延续。最后被执行的延续线圈十逻辑通流的状态。它将用于下个被执行的延续触点。如果逻辑在执行一个延续触点之前未执行一延续线圈，则触点的状态不是通流状态，当PLC从STOP转换到RUN状态时，延续触点的状态不是通流并且到RUN模式时，延续线圈置位时才通流。.

每个回路只能有一个延续线圈和延续触点，延续触点必须放在第1列，而延续线圈只能放在第10列。如下例所示：



Chapter 5

第五章　定时器和计数器

本节讲述了如何应用接通延时和停止型定时器和减计数器。与这些功能模块相关的数据时通过电源循环来保持住的。

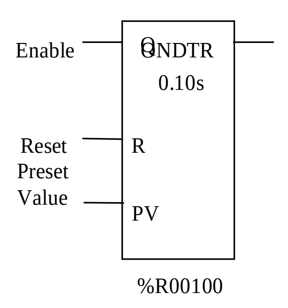
缩写	功能	Page
ONDTR	保持接通延时定时器	5-3
TMR	简单接通延时定时器	5-5
OFDT	断开延时定时器	5-8
UPCTR	加计数器	5-11
DNCTR	减计数器	5-13

定时器和计数器功能模块所需的数据

每个定时器或计数器用%R 存贮器和三个字（寄存器）来贮存如下信息：

当前值(CV)	字 1
预置值 (PV)	字 2
控制字	字 3

当加入一个定时器或计数器时，必须对这三个字（寄存器）直接加入一个起始地址，在图示功能模块的下面。例如：这个块的起始地址是%R00100.



注意

对于3个字的定时器/计数器模块不要使用连续寄存器。当寄存器模块产生重叠时，Logicmaster 不会检验操作或报警。如果用户把模块当前值置于前一个模块的预置值之上，则定时器和计数器会不执行。

控制字将储存其与功能模块相关的布尔（逻辑）输入和输出的状态，如下面的格式所示：

15 14 13 12 11 10 9 8 | 7 6 5 4 3 2 1 0

Reserved

Reset input

Enable input, previous execution

Q (counter/timer status output)

EN (enable input)

Reserved

0～11位用于定时器精度，对计数器，0～11位则不用。

注意

如果对于功能的PV（预置值）输入参数与3个字块的第二个字使用相同的地址时应当注意，如果PV不是一个常量，PV输入一般是与第二个字不同存贮区域的地址。一些程序选择第二个字作为PV输入，如，当起始地址为%R0101的3个字块时使用%R0102作为PV,当定时器或计数器运行的时候请求允许修改PV，请求能够读取第一个字（CV）或第三个字（控制字），但是请求不能修改这些值，因为如果它们被写入时，功能块将不工作。

在某些功能块时特殊注意

当使用Bit Test, Bit Set, Bit Clear 或位功能模块时, 位的数量从1到16, 不是0到15，如上面显示。.

ONDTR

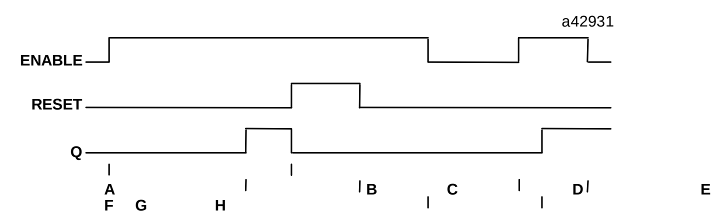
保持接通延时计时器（ONDTR）,当其接收电流时，将增加计时，而在停止时则保持其计时值。可以十分之一秒（缺省选择）或百分之一秒或千分之一秒进行计时，范围为0至＋32767时间单位。这类计时器的状态可在电源故障时被记忆，在电源接通时，不会出现自动清除。

每当ONDTR第一次通电时，它便开始累计时间（当前值），当这类定时器被接入梯形逻辑中时，其当前值便被更新。

注意

> 如果在一个CPU扫描期间，允许具有同样参考地址的相同定时器多次出现，则时间的当前值将相同。

当前时值等于或超过预置值PV时，输出Q有电流通过。只要定时器继续接收电流，它就继续累计时间，直到达最大值为止。一旦达到最大值，则无论使能（EN）输入的状态如何，最大值都将被保持住，而输出Q则仍有电流通过。



A＝ 使能（ENABLE）升高，计时器开始累计时间.

B＝ 当前值达到预置值 PV; Q 升高.

C＝ 复位RESET升高; Q下降，累计时间复位.

D＝ 复位RESET下降; 定时器又开始累计时间。

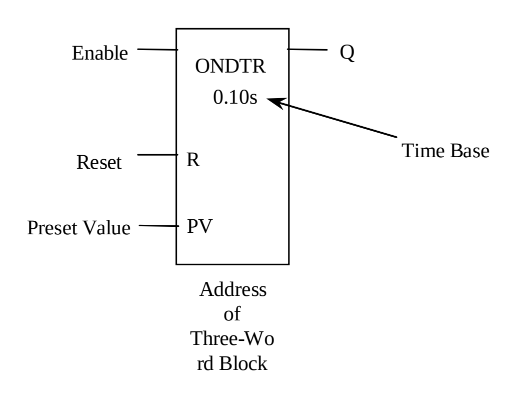
E＝使能ENABLE 下降; 计时器停止累计. 所累计的时间停在同一值.

F＝使能ENABLE再升高; 定时器继续累计时间.

G＝当前值变成等于预置值PV; Q升高. 定时器继续累计时间，直到使能（ENABLE）下降或复位（RESET）升高，或当前值变为最大值为止。

H＝使能ENABLE下降; 定时器停止累计时间.

当定时器停止通流时，当前值将停止增长并被保持住。输出Q（如果有电流）则仍继续通流。当功能模块再次接通电流时，当前值将再从原保持值开始继续增长。当复位（RESET）接通电流时，当前值置回零，而输出Q停止通流。在 35x, 36x, 和 37x 系列 PLC中,如果ONDTR的使能没有, PV = 0 并且复位没有通流，则输出没有通流. 然而，在 311–341型 PLC中，在相同条件下，输出则通流.



参数

参数	说 明
地址（address）	ONDTR可使用%R 存贮器的3个连续字（寄存器）来贮存下列数据: - 当前值 (CV) = 字1. - 预置值 (PV) = 字 2. - 控制字 = 字 3. 当进入ONDTR时，必须键入模块图底部的3个连续字（寄存器）的位置地址。 **注意:** 不得使用带其他指令的该地址。 重叠参考地址会引起定时器误操作。
使能	当ENABLE通流时，定时器的当前值便增长.
R	当R通流时，定时器将当前值复位为零
PV	PV是定时器使能或复位时，复制到定时器预置值的数据
Q	当当前值大于或等于预置值时，输出 Q通流
时间	时间增量为0.1,0.01,0.001秒，用于低位PV预置和CV值

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
address								•				
enable	•											
R	•											
PV		•	•	•	•		•	•	•	•	•	•
Q	•											•

- 当电流流过功能块时的有效参考地址或位置.

示例

在下例中，一个记忆接通延时定时器用来确立一个信号 (%Q0011) ，后者在%Q0010接通后8秒后接通而在 %Q0010断开后断开,因为当 %Q0010断开时,就好比一般触点被输入复位一样。8秒的时间时预置值80与时间0.1s的积。

```
|
| %Q0010 |      |                                                            %Q0011
|——| |———|ONDTR|——————————————————————————————————————————————————————————————( )—
|        | 0.1s|
| %Q0010 |      |
|——|/|———|R     |
|        |      |
|        |      |
| CONST —|PV    |
| +00080 |___ _|
|
|       %R0004
```

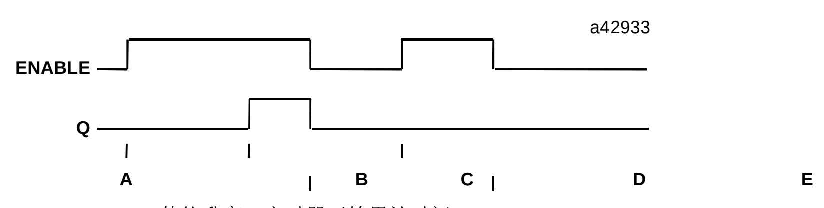
TMR

简单的接通延时定时器（TMR），通电时计时，而停电时复零。
可以按十分之一秒或百分之一或千分之一进行计时，范围时0～＋32767时间单位。
当TMR通流时，定时器百开始累计时间（当前值），当遇到反映总消逝时间的逻辑时，当前值被更新，其中定时器从最后一次复位起被允许计时。

注意

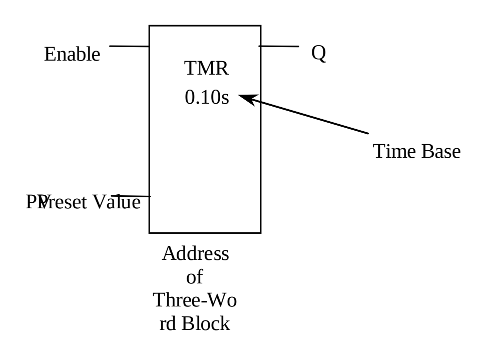
> 在一个CPU扫描期间,如果允许具有相同参考地址的相同定时器多次出现，则计时的当前值将也相同。

只要使能逻辑保持位ON，计时值被更新。当当前值等于或超出预置值PV时，则功能模块便开始对右侧通过电流。计时器继续累计时间直至达到最大值位为止。当使能参考参数由ON转换为OFF时，定时器则停止累计时间，而当前值复零。TMR功能的模块状态对电源故障时可记忆的。当接通电源时，不会出现自动回零（清除）。



A = ENABLE使能升高，定时器开始累计时间
B = 当前值达到预置值PV，Q升高，定时器则继续累计时间

C = ENABLE下将，Q下降，定时器停止累计时间，而当前值被清除
D = ENABLE升高，定时器开始累计时间
E =ENABLE在当前值达到预置值PV之前下降，Q保持低，定时器停止累计时间，并被清零



参数

参数	说明
地址（address）	TMR可使用％R存贮器的3个连续字（寄存器）来贮存下列数据，内容：: 当前值 (CV) = 字 1. 预置值 (PV) =字2. 控制字 =字 3. 若需进入TMR，必须键入功能图形下方的3个连续字（寄存器）的位置地址. **注意:** 不得使用带其他指令的该地址。重叠参考地址会引起定时器误操作。
使能	当ENABLE通流时，定时器的当前值便增长.
PV	预置值输入.当定时器使能或复位时，PV是复制到定时器当前值里的值.，当时间达到PV值时，定时器将输出Q置为ON。
Q	当TMR被使能而当前值大于或等于预置值时，输出Q便通流。
时间	定义为十分之一秒、百分之一秒和千分之一秒，时间与预置值的积来作为有效的时间

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
address								•				
enable	•											
PV		•	•	•	•		•	•	•	•	•	•
Q	•											•

• 可通流功能模块的有效参考地址与位置

示例

在下例中，一个接通延时定时器TMRID用来控制线圈DWELL接通时间的长度。当常开（瞬动）触点DO-DWL接通时，线圈DWELL通电。线圈DWELL的触点将保持线圈DWELL被通电（当触点D0-WLL释放时），而且也启动定时器TMRID。当TMRID达到其预置值的一半时，线圈REL通电，中断线圈DWELL阻通状态。触点DWELL中断对TMRID的通电，后者复位其当前值，且使线圈REL断电。然后线路就准备触点DO-DWL的另一瞬动动作。.

```
|
| DO_DWL   REL                                                         DWELL
|——|  |——+——|/|——————————————————————————————————————————————————————————( )—
|        |
| DWELL  |
|——|  |——+
|         _____
| DWELL  |     |                                                         REL
|——|  |——| TMR |—————————————————————————————————————————————————————————( )—
|        | 0.1s|
|        |     |
| CONST —|PV   |
| +00005 |_____|
|
|       TMRID
```

OFDT

当电流断开时，断开延时定时器增值，并且在电流接通时它复位至0。时间以十分之一秒，百分之一秒或千分之一秒计数，此范围为0～＋32767个时间单位。此定时器的状态当电源断开时为保留值，在电源接通时亦不出现回零。

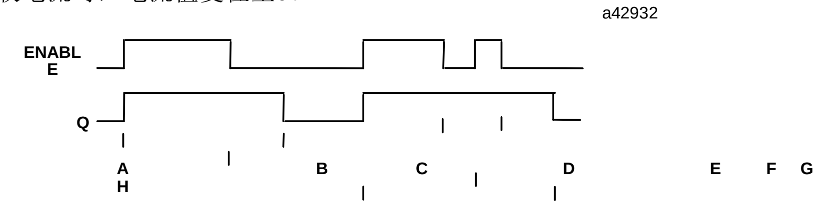
当OFDT首次接收电流时，即把电流向右传送，并且当前值（CV）被置位为0.（OFDT使第一个字（寄存器）作为它的CV存贮器位－－参看“参数”；在下一页上有附加的信息）。只要此功能模块接收到电流，输出就保留。如果此功能模块停止接收来自左方的电流，那么它会继续向右输送电流，并且定时器开始在当前值内累积时间。

注意

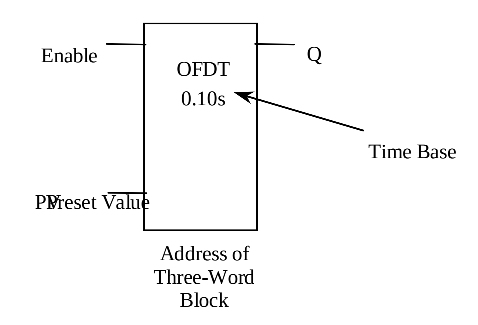
> 如果在一次CPU扫描期间，带有同样参考地址的同一定时器多次出现的话，那么定时器的当前值相同

如果预设定值为零或非，那么OFDT就不传送电流。
每次调用功能模块使逻辑置于OFF时，当前值即被刷新，以反映由于定时器关闭的逝去时间。当当前值(CV)等于预设定置（PV）时，此功能就停止向右传送电流。当这现象出现时，定时器就停止累积时间－－参看下面的C部分。

当此功能再次接收电流时，电流值复位至0。



A = ENABLE 和Q 同时升高 ; t定时器复位 (CV = 0).
B = ENABLE下降; 定时器开始累积时间.
C = CV达到 PV值; Q 下降，定时器停止累积时间
. D = ENABLE升高; 定时器复位 (CV = 0), Q升高.
E = ENABLE下降; 定时器开始累积时间, Q 升高
F = ENABLE升高; 定时器复位(CV = 0), Q 升高h.
G = ENABLE下降; 定时器开始累积试讲, Q升高.
H = CV达到 PV值; Q 下降,定时器停止累积时间.



当OFDT用于一个不是每次扫描都调用的程序块时，定时器可累加程序块的调用时间，直到定时器复位为止。此意即为：其作用类似于一个定时器以低于在主程序块中更慢些的扫描在一个程序中操作。对于长时间处于非激活态的程序块，定时器应在编程时考虑这一功能。例如：如果程序块中的某个定时器复位且程序块未调用（即非激活态）为4分钟，当程序块被调用时，此4分钟即已累加进去。启动定时器时该时间即被用于定时器，除非定时器为首此复位。

参数

参数	说明
地址（address）	OFDT使用%R 存贮器的3个连续字（寄存器）来贮存下列数据： 当前值 (CV) = 字 1. 预置值 (PV) = 字2. 控制字 = 字3. 若需进入OFDT,必须在代表功能块的图形正下方输入这3个连续字（寄存器）的位置地址 。 **注意:** 不得使用带其它指令的该地址。 重叠参考地址会引起定时器误操作
使能	当enable通流时，定时器的当前值便增加。 (CV)达到 (PV), 定时器停止计时，Q输出关断.
PV	预置值输入.当定时器使能或复位时，PV是复制到定时器设定值里的值.，当时间达到PV值时，定时器将输出Q置为ON。
Q	输出Q产生的条件（1）当使能输入为ON（2）在使能输入变为OFF以后，当前值(CV)小于设定值(PV)。
时间基准	这个参数增量可以编辑为十分之一或百分之一或千分之一秒，时间基准值的增量由预置值（PV）输入参数中的数量决定的。

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
address								•				
enable	•											
PV	•	•	•	•	•		•	•	•	•	•	•
Q	•											•

• 可通流功能模块的有效参考地址与位置

示例

在下例中，OFDT定时器用于关闭一输出线圈%Q0001，一输入%I0001无论何时打开。在%I0001开后此输出开2s

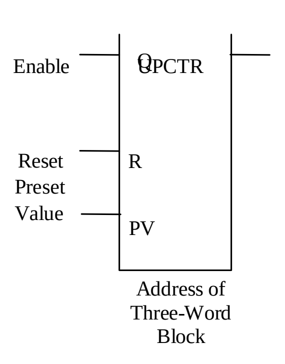
%I0001
%Q0001
OFDT
0.10s
CONST
+00020
PV
%R0019

在下例中，输出线圈使用的是求反线圈，在这个回路中，只要触点%I0001关闭，OFDT定时器将方向关闭输出线圈%Q0001，在 %I0001 打开, %Q0001 持续2秒钟OFF状态后变为ON。

%I0001
%Q0001
OFDT
0.10s
CONST
+00020
PV
%R0019

UPCTR

递增（加）计数器用来结算一个指定值，计数范围是0～＋32767。当递增计数器的复位端为ON时，计数器的当前值便复位为零。每次使能端输入从OFF转换到ON，则当前值增1.当前可递增超过预置值PV。每当当前值大于或等于预置值时，输出便为ON。断电时。UPCTR的状态可在电源故障下记忆住；在电源接通时，并不自动清除。



参数

参数	说明
地址	UPCTR使用%R存贮器的3个连续字（寄存器）来贮存下列数据 当前值 (CV) = 字 1. 预置值 (PV) = 字2. 控制字 = 字3. 若需进入UPCTR,必须在代表功能块的图形正下方输入这3个连续字（寄存器）的位置地址 。 **注意:** 不得使用带其它指令的该地址。 重叠参考地址会引起定时器误操作
使能	在enable的正向变换下，当前数值增1.
PV	预置值输入.当计数器使能或复位时，PV是复制到计数器设定值里的值.，当计数到PV值时，计数器将输出Q置为ON，如果预置值是常量，它的数值必须在0和32767之间。
Q	当前值大于或等于预置值时，输出Q便通流。
R	复位输入. 当R为ON是，当前值(CV), 被复位为0。

有效存贮器类型

Parameter	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
address								•				
enable	•											
R	•											
PV		•	•	•	•		•	•	•	•	•	•
Q	•											•

• 可通流功能模块的有效参考地址与位置

示例

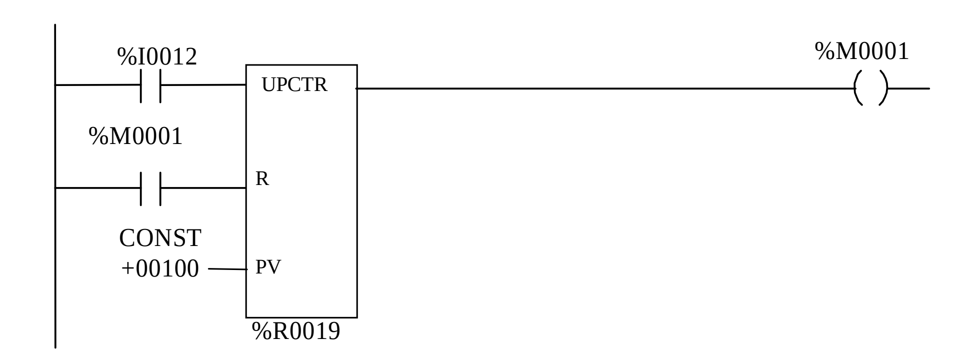
基本计数器回路

在下例中，输入%I0001在每次有OFF转为ON时，加计数器UPCTR当前值CV就计数加1，预置值设定为100，当计数器加到100时，线圈%Q0001就会接通，计数器将一直计数下去，直到超过预置值到它的最大值（32767）或由%I0020接通关断计数器才重新计数。.在当前值等于或大于预置值时 %Q0001就会被接通。

%I0001
UPCTR
%Q0001
%I0020
R
CONST
+00100
PV
%R0019

自复位计数器

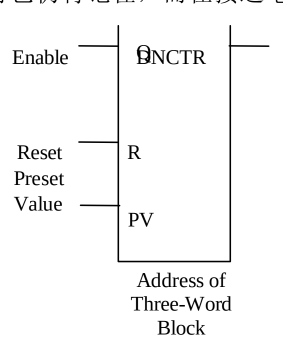
在下例中，输入%I0012 每次由OFF转为ON时，加计数器 UPCTR就会计数加1，每当百分之一被计数，内部线圈 %M0001便接通，每当%M0001为ON状态时，则累计的计数器便复位为零。



DNCTR

减计数器(DNCTR) 功能模块是用来从一个预置值递减计数。最效的预置值为零，最大的预置值为＋32767。最小的当前值为－32768，当复位时，计数器的当前值被设定为计数器的预置值，当输入使能从OFF转为ON时，当前值就减其，当当前值等于零或小于零时，输出便接通。

DNCTR的当前值遇到电源故障时也仍将记住，而在接通电源时，并不自动清除。.



参数

参数	说明
地址	DNCTR使用%R存贮器的3个连续字（寄存器）来贮存下列数据 当前值 (CV) = 字 1. 预置值 (PV) = 字2. 控制字 = 字3. 若需进入DNCTR,必须在代表功能块的图形正下方输入这3个连续字（寄存器）的位置地址 。 **注意:** 不得使用带其它指令的该地址。 重叠参考地址会引起定时器误操作
使能	在使能（enable）端正向变换时，当前值减1
PV	当计数器在使能或复位状态时，PV时复制到计数器预置值的值
Q.	当当前值小于或等于零时，输出Q通流
R	复位输入，当R端通流时，当前值CV复位为（PV），并且输出Q没有通流。

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
address								•				
enable	•											
R	•											
PV		•	•	•	•		•	•	•	•	•	•
Q	•											•

• 可通流功能模块的有效参考地址与位置

示例

在下例中，被定义COUNTP的减计数器在输出%Q0005接通前，将计数5000新等分。

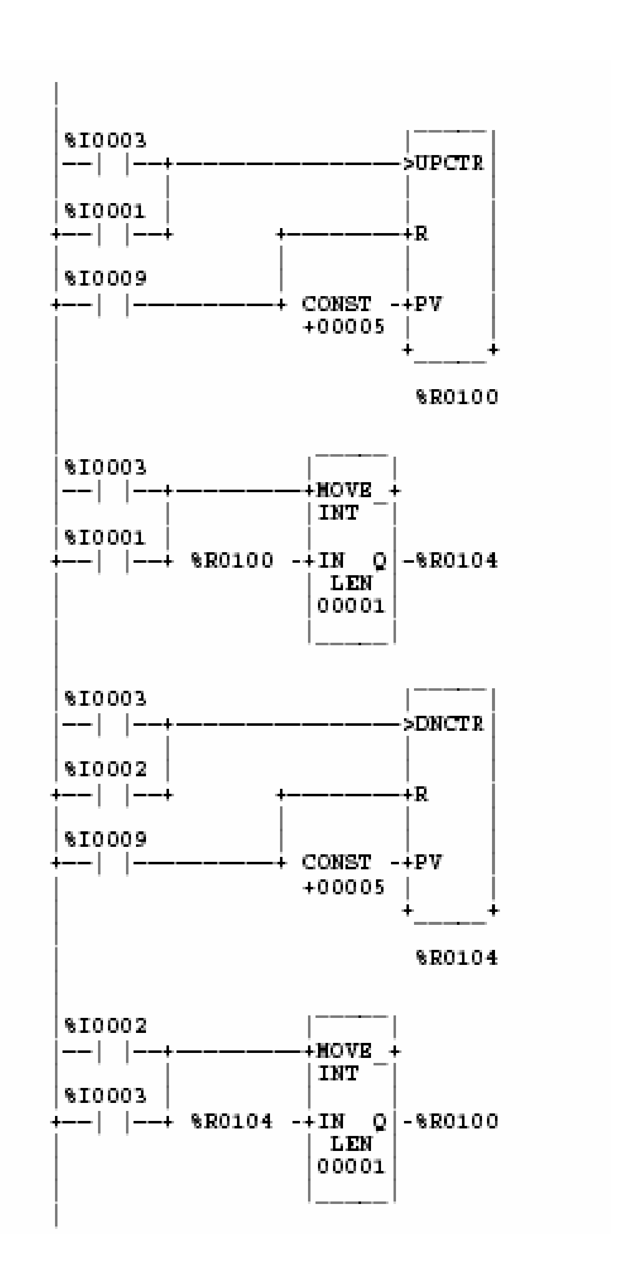
```
|          _____
|NEW_PRT  |     |                                                                 %Q0005
|--|  |-- >DNCTR|------------------------------------------------------------------( )-
|         |     |
|NXT_BAT  |     |
|--|  |---|R    |
|         |     |
|         |     |
| CONST  -|PV   |
| +05000  |     |
|         |_____|
|          COUNTP
|
```

详细计数示例

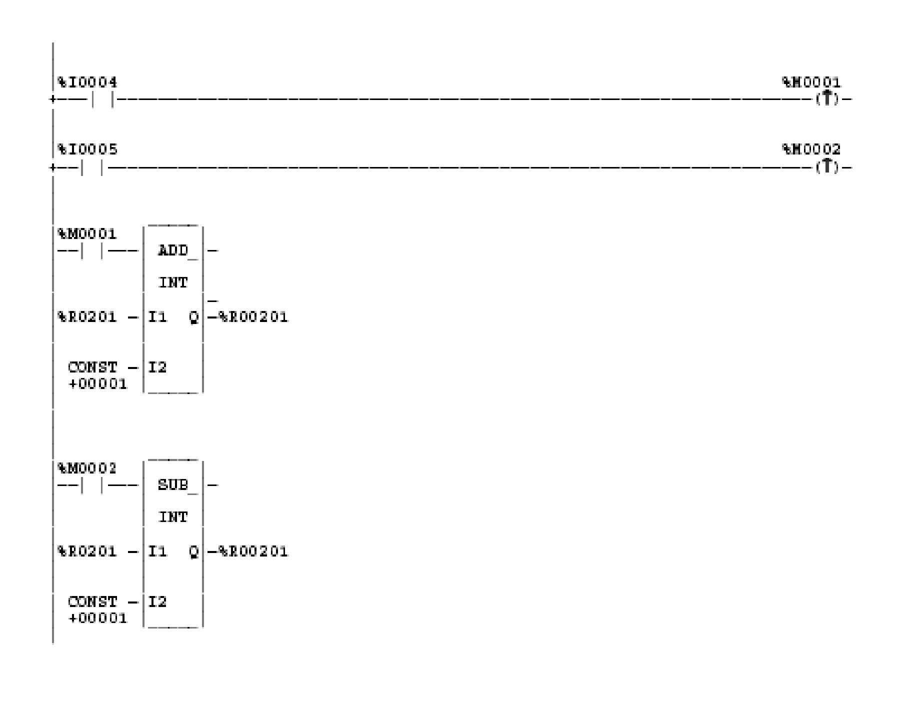
在下例中，PLC用来记住在暂时存贮器内所保存的等分数。那是利用90-30/20/ Micro 指令系统以完成该功能的两种方法。

第一种方法是使用一个带一个用于累加值或当前值的分配寄存器的加/减计数器对，当等分进入存贮器时，加计数加1，存贮区的等分当前值也增加一个1的数值。每当一等分离开存贮区，减计数器减1，存贮值减1。为避免与分配寄存器冲突，这两个计数器使用不同的寄存器地址，但分配相同的当前值（CV）地址，对这种应用来说，这是优先选用的方法。

在下例中%I0001为加计数t, %I0002 减计数, %I0009复位计数为零和%I0003,当通电保持时，计数器就像%I0001和%I0002接通时. 计数器的值从％R0100读取



如下所示第二种方法，将利用ADD和SUB功能来提供存贮统调。%R00201是通用寄存器. 当计数增加 ($I0004接通), ADD功能将在%R00201加数. 当计数减少 (%I0005 接通), SUB 功能将在%R00201地址中减数. 既然如此, 转换线圈可作为ADD和SUB功能的脉冲使能输入，如果使能输入是脉冲形式，ADD和SUB功能在使能时的扫描中只执行一次（UPDTR和DNCTR功能不需要转换线圈，因为他们的使能输入在转换功能块中已经成立）第六章将详细介绍关于ADD和SUB功能块。



Chapter 6 第六章 数学运算功能模块

这一节将介绍系列 90-30/20/Micro 的数学预算功能模块：

缩写	功能	说明	页码
ADD	加法	两个数相加	6-2
SUB	减法	一个数减另外一个数	6-2
MUL	乘法	两个数相乘	6-2
DIV	除法	一个数被另一个数除得商	6-2
MOD	模除	一个数被另一个数除，得余数	6-7
SQRT	平方根	求一个整数的平方根	6-9
SIN, COS, TAN, ASIN, ACOS, ATAN	三角函数 †	执行输入IN实数值的相应功能	6-11
LOG, LN EXP, EXPT	对数/指数，函数 †	执行输入IN实数值的相应功能	6-13
RAD, DEG	弧度转换 †	执行输入IN实数值的相应功能	6-15

† 三角函数、对数函数及弧度转换功能仅使用于35x和36x系列 CPU, 9.00或更高的版本,并不是所有 CPU352 和 CPU37x的版本.

注意

除法和模除功能相近，输出不同。除法得到商，而模除为余数。

标准数学运算功能模块(ADD, SUB, MUL, DIV)

数学运算功能模块包括加、减、乘、除。当一个功能模块接收电源是，依据输入参数I1和I2来完成相应的数学运算功能。这些参数必须具有相同的数据类型。输出Q数据类型与I1和I2相同。

数学运算法则

结果的符号	对于带符号数字算法的标准数学准则的应用决定结果的符号。
加法	加法指令运用公式 I1 + I2 = Q.
减法	减法指令运用公式 I1 – I2 = Q**.**
乘法	乘法指令运用公式 I1 x I2 = Q.
除法	除法指令运用公式 I1 ⎮I2 = Q. **INT 和DINT类型**. DIV对于INT或DINT类型圆整到完整数的商（剩余的被舍弃）; 并不是圆整到最接近的整数. 例如 53 除以 5等于 10 (余数3 被舍弃). **对于REAL类型**. DIV 得到REAL类型的十进制结果。
模除	模除指令仅适用INT和DINT类型（不支持REAL类型）， MOD指令使用的公式是I1 ⎮I2 = Q. 然而, MOD得到的仅是除法操作中舍弃的余数，例如， 53 除以 5 等于3 (商10被舍弃).

数学运算的数据类型

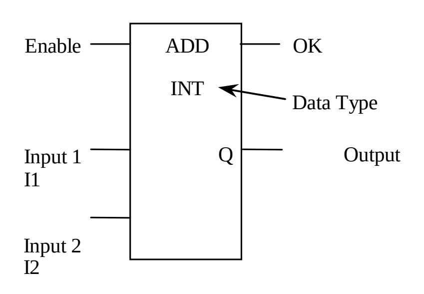
在编辑一个数学功能模块后，需要选择数据类型，这个数据类型出现在功能块名称的下面（如下图），数学功能块用到三种数据类型，下表所示

数据类型	说明
INT	带符号整型.
DINT	双精度带符号整型
REAL*	浮点数t

*REAL数据仅使用35x和 36x系列 CPU, 9.00或更高版本,并不支持说有CPU352 和 CPU37X版本

缺省数据类型是带符号整数，然而，它能在选定功能模块之后加以改变。关于数据类型的更多信息，请参考第二章第二节“程序组织和用户参考标号/数据”

如果INT或DINT运算结果产生溢出，则输出参考标号将被值成该数据类型的最大可能值。对于带符号的数字，符号则标出溢出的方向。如果运算结果未溢出，则OK输出置为ON，否则，置为OFF。如果使用带符号或DINT整型，则DIV和MUL功能的结果符号取决于I1和I2的符号。



参数

参数	说明
使能enable	当 功 能 模 块 允 许 操 作 （ 使 能 ） 时， 便 执 行 运 算
I1	I1 即运算中所用的第一个值为一常量或参考值（I1是在算术等式的左边，如 I1-I2）
I2	I2即运算中所用的第二个值为一常量或参考值（I1是在算术等式的右边，如 I1-I2）
ok	当功能完成而无溢出，除非出现无效运算，则OK输出端便接通。
Q	输出Q即运算的结果

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
I1		o	o	o	o		o	•	•	•	•†	
I2		o	o	o	o		o	•	•	•	•†	
ok	•											•
Q		o	o	o	o		o	•	•	•		

• 电流经功能模块时的有效参考（地址）或地点

o 仅用于 INT 数据的有效参考地址，对DINT或REAL则无效;

† 用于双精度带符号整数，数值限制在－32767~＋32767之间的常数。

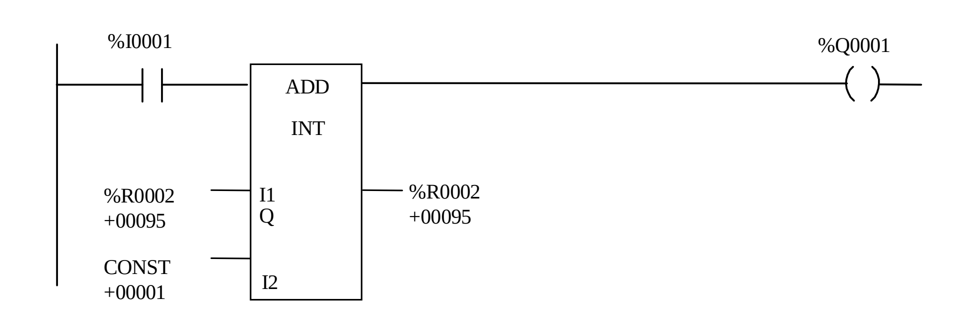
注意

缺省类型时16位或单一的寄存器操作数的INT。按F10即可改变类型选择DINT，32位双字，或REAL（仅适于 35x, 36x, 和 37x 系列 CPU）。PLC INT值占据一单一的16位寄存器，％R,%AI或%AQ，DINT值需要两个连续寄存器，这两个连续寄存器在第一字中有低位和在第二字中带有标有正负号的上16位。REAL值仅在35x和 36x系列 CPU (9.00或更高版本) 和在CPU352和CPU37x, 所有的版本，也占据一 32位双积存器，此寄存器带有指数和尾数紧随的带符号高位。

数学运算功能模块示例

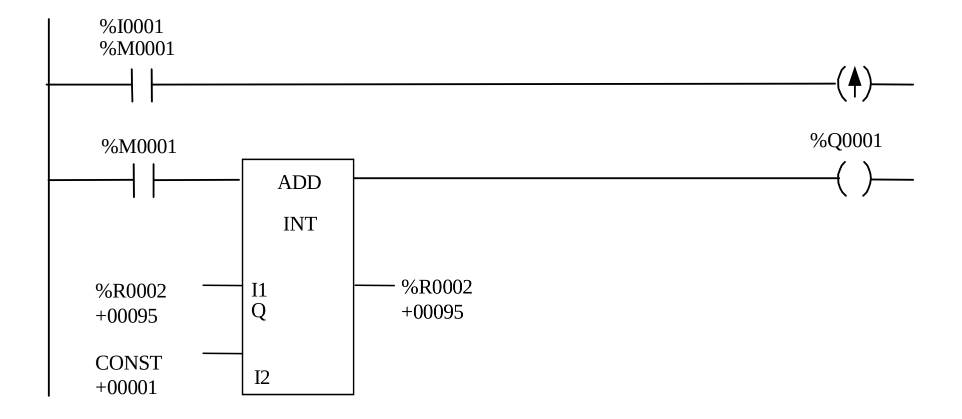
程序的ADD回路

在下例中，试图创建计数开关%I0001闭合的次数的计数器。运行的总述存贮在寄存器%R0002中，这个设计的目的是当%I0001闭合时，ADD指令在%R0002（输入I2）中加1并且将新值替换到右边%R0002（Q输出），这个程序在%I0001闭合时PLC每个扫描周期ADD指令将执行一次。所以，例如，如果%I0001停留5个扫描周期，输出将增加5次，即便%I0001在这个时期中闭合一次。正确处理这个问题时，ADD指令的使能输入应该来自跳转（一次脉冲）线圈，像下面第二个图。



在下面改良的回路中，%I0001输入开关控制跳转（脉冲）线圈%M0001，
每次触点%I0001闭合，ADD功能使能输入的触点%M0001为ON的扫描周期只一次,为了再次%M0001触点闭合，触点%I0001只有再次接通断开.

正确ADD回路设计



数学功能及数据类型

功能	操作	显示
ADD INT	Q(16位) = I1(16位) + I2(16位)	带有正负号的10进制的5位数
ADD DINT	Q(32 bit) = I1(32 位) + I2(32位)	带有正负号的10进制的8位数
ADD REAL*	Q(32 bit) = I1(32位) + I2(32位)	带有正负号和小数的10进制的7位数
SUB INT	Q(16位) = I1(16位) – I2(16位)	带有正负号的10进制的5位数
SUB DINT	Q(32位) = I1(32位) – I2(32位)	带有正负号的10进制的8位数
SUB REAL*	Q(32位) = I1(32位) – I2(32位)	带有正负号和小数的10进制的7位数
MUL INT	Q(16位) = I1(16位) * I2(16位)	带有正负号的10进制的5位数
MUL DINT	Q(32位) = I1(32位) * I2(32位)	带有正负号的10进制的8位数
MUL REAL*	Q(32位) = I1(32位) * I2(32位t)	带有正负号和小数的10进制的7位数
DIV INT	Q(16位) = I1(16位) / I2(16位)	带有正负号的10进制的5位数
DIV DINT	Q(32位) = I1(32位) / I2(32位)	带有正负号的10进制的8位数
DIV REAL*	Q(32位) = I1(32位) / I2(32位)	带有正负号和小数的10进制的7位数

* 仅35x 和 36x 系列 CPU, 9 或更高版本, 和 CPU352 和CPU37x的所有版本.

注意

输入和输出数据类型必须相同。MUL和DIV功能模块不支持系列90-70 PLC所做的混合模式如：2个16位的士儒的MUL_INT得出一16位结果，而不是－32位。使用MUL DINT32位结果需要两个输入都是32位。当DIV DINT将－32位I1除以32位得I2得到－32位结果时，DIV INT除以－16位的I2得到－16位结果。

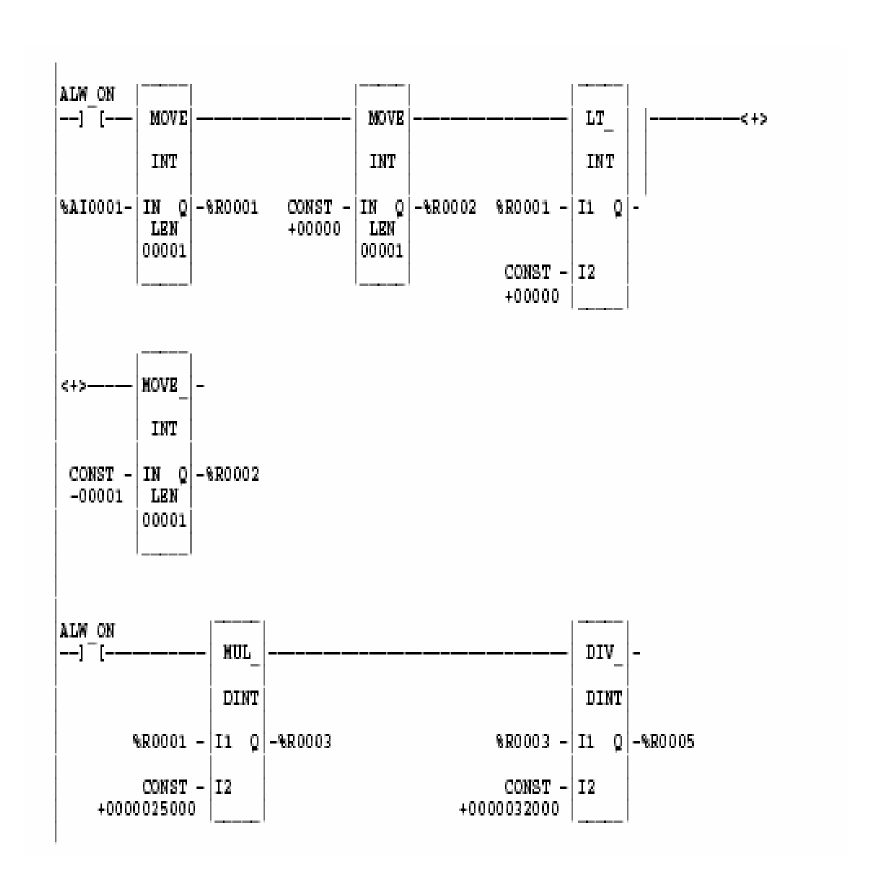
如果无数学溢出，那么这些功能可能通过电流。如果出现溢出，那么结果就是带有相应正负号的最大值，且没有电流。

在使用MUL和DIV功能时相应小心注意避免溢出。如果你不得已把INT值转换至DINT值，记住CPU使用带有正负号的标准2's 补码，其延伸至第二个字的最高位。必须检验低16位字的正负号，并将它延伸至第二位16字。如果在－16位INT字中最有效的位为0（正）则把0送至第2字。如果－16为字中的最有效的位为－1（负），则传送－1或16进制OFFFFh至第2个字。如果低16位字（第一寄存器）是－DINT32位字的INT部分，则把DINT转换成INT就容易一些，高16位或第2个字应是0（正）或－1（负），否则DINT数太大不能转换成16位。

示例

一般程序通过MUL运算和随后的DIV 和ADD 运算对模拟量输入数值进行换算. 0到 ±10 电压模拟量输入对应%AI输入寄存器数值为0 到 ±32,000。使用 INT MUL 功能乘以该输入寄存器将导致溢出因为INT 类型指令输入和输出范围是 32,767 到–32,768。使用 %AI数值作为MUL DINT 输入也不能正常运算因为32-位 I1 同时包含二个模拟量输入. 为了解决这个问题, 你可以将模拟量输入移入双字低位，然后测试符号位设置第二个寄存器为0如果符号位正的，如果为负的设置为－1. 然后使用MUL DINT得到32位的双精度运算结果, 其结果用于随后的DINT DIV 功能.

例如, 以下逻辑用来转换± 10电压输入%AI1到 ±25000工程单位到%R5.



一种低精度编程方法是首先进行DIV运算，且使用INT类型，然后执行MUL指令。DIV指令I2数值是32, MUL指令I2数值是25.　这样可保持缩放比例且保持数值在INT类型范围。然而，DIV指令会丢弃余数, 因此当DIV输出结果进行MUL运算, 由于丢弃余数导致的误差进行乘法运算. 转换后结果和输入非线性且小于输入.

通过对比, 上例中, 结果更精确因为DIV运算最后执行, 因此丢弃的余数不进行乘法. 即使需要更高精度, 本例中采用REAL类型数学指令这样余数不会丢弃.

MOD (INT, DINT)

模除（求余）功能模块用于相同数据类型的一个数除以另一个数，以求得余数。结果的符号总是与输入参数I1的符号相同。
MOD 功能模块将按下列数据类型进行运算：

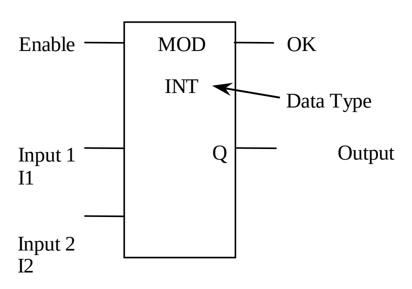
数据类型	说 明
INT	带符号整数.
DINT	双精度带符号整数

缺省数据类型是带符号整数，然而，它在选定功能模块后加以改变。关于数据类型的更多信息，请参考第二章第二节“程序组织和用户参考标号/数据”。
当功能模块接受电流时，输入参数I1将被参数I2除，这些参数必须是相同的数据类型。输出Q则用下式计算。

```
Q = I1 - ([I1 DIV I2] * I2)
```

其中，DIV运算将产生一个整型数，Q和输入参数I1、I2有相同的数据类型。

当功能模块接电时，OK端总置为ON，除非试图除以0，在这种情况下，OK端置OFF.



参数

参数	说明
使能Enable	当功能模块被允许操作，便执行运算。
I1	I1即为除以 I2值的常数或参考地址。.
I2	I2 对除 I1的值保持为常数或参考地址。.
OK	当功能模块完成而无溢出时，则OK输出接通。
Q	输出Q即为I1被I2除的结果余数。.

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
I1		o	o	o	o		o	•	•	•	•†	
I2		o	o	o	o		o	•	•	•	•†	
ok	•											•
Q		o	o	o	o		o	•	•	•		

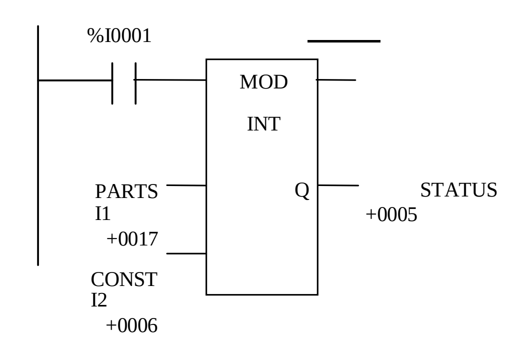
• 电流经功能模块时的有效参考（地址）或地点

. o 仅用于INT数据的有效参考地址，对DINT则无效。.

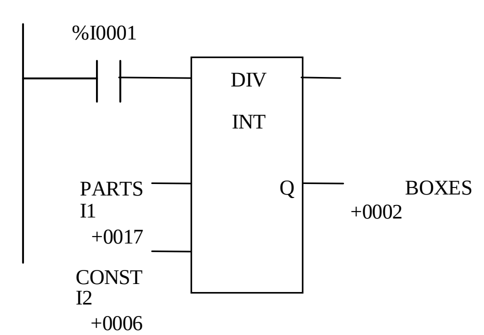
† 常数，对于双精度带符号整数，数值限制在－32767～＋32767之间的常数。

示例

在下例中, MOD使能为ON, the MOD功能寄存器(PARTS)除以6取余. 输出 (STATUS) 表示二者相除余数（1 ～ 5). 如果二者能整除,输出Q为0; 如果二者不能整除, 输出Q为二者相除余数. 本例中17除以6，商为2，余数为5。



确认boxes数值, 你可以使用 DIV指令.



一个可能的问题是寄存器PARTS最大32,767 .如果你需要更大输入, 需要增加一些逻辑(1)复位 PARTS 寄存器在它达到最大值之前, (2) 复位 PARTS 寄存器之前记录boxes数值(3) 复位寄存器PARTS当STATUS 寄存器是0因此它的计算仍精确.

SQRT (INT, DINT, REAL)

平方根运算 (SQRT)的功能模块用于求一个数值的平方根，当该功能模块接收电流时，输出Q的值时输入IN平方根的整数部分。输出Q必须和IN具有相同的数据类型。
求平方根功能模块按下列数据类型运算。

数据	说明
INT	带符号整数
DINT	双精度带符号整数.
REAL	浮点数

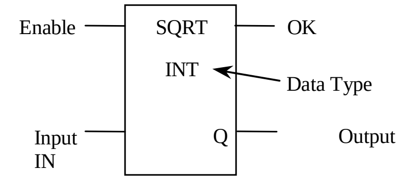
对于 INT和 DINT型，仅仅是平方根的整数部分被输出t.小数部分被舍弃。 例如：2或3的平方根是1；5、6、7或8的平方根是2。

注意

REAL数据类型仅使用于 35x和 36x 系列的 CPU, 9.00或更高的版本，和 CPU352和 CPU37x.的所有版本

缺省数据类型是带符号的整数，但它可在选定功能模块之后改变。关于数据类型的更多信息，请参考第二章第二节“程序组织及用户参考标号/数据”。
如果功能模块完成运算而无溢出时，则OK端置ON，除非出现下面的非法运算，OK端置OFF.：

- IN < 0
- IN是 NaN (非数值)



参数

参数	说明
使能enable	但功能模块允许操作，便执行运算。
IN	IN 将为所求平方根值保持一常数或参考地址，IN小于零，则功能块无电流溢出。
ok	当完成运算无溢出，则输出被接通，除非出现非法运算。
Q	输出Q即为IN的平方根值。

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable		•										
IN		o	o	o	o		o	•	•	•	•†	
ok	•											•
Q		o	o	o	o		o	•	•	•		

• 电流经功能模块时的有效参考（地址）或地点

. o 仅用于INT数据的有效参考地址，对DINT则无效。.

† 常数，对于双精度带符号整数，数值限制在－32767～＋32767之间的常数。

示例

在下例中，只要%I0001接通为ON时，整数%R0008的平方根值将放在地址%R0009中。

%I0001

OK SQRT

INT

%R0008 IN +00019

Q

%R0009 +00004

根据计算精度要求，可使用REAL-类型SQRT指令，如下图所示精度更高.

%I0001

OK SQRT

REAL

%R0008 IN +19.00000

Q

%R0009 +4.358899

三角函数 (SIN, COS, TAN, ASIN, ACOS, ATAN)

SIN, COS, 和 TAN功能用于得出输入相应的三角函数正弦、余弦和正切值。当这些功能之一接收到电流时，它就计算IN的正弦（余弦和正切），它的单位时弧度，并且结果存入输出Q内，IN和1都是浮点值。

ASIN、ACOS 和ATAN功能用于得出与输入相应的反正弦，反余弦和反正切。当这些功能中的一个接收到电流时，它就计算IN的反正弦（反余弦或反正切），并且结果存入输出Q内，它的单位时弧度，IN和Q都是浮点值。

SIN, COS,和 TAN功能接受输入值的范围为：
$-2^{63} < IN < +2^{63}$, (2^{63} H $9.22x10^{18}$).

ASIN 和ACOS功能接受输入值的范围为较窄： – 1 < IN < 1. 给IN参数一有效值，ASIN_REAL功能将得到一结果Q如下：

ASIN (IN) δ – $\frac{}{2}$ δQ

ACOS_REAL功能将得到一结果Q如下:

ACOS (IN) = 0 δQ

ATAN功能接受输入值的最宽的范围：– δ IN δ + . 给IN参数一有效值， ASIN_REAL功能将得到一结果Q如下：

ATAN (IN) = – $\frac{}{2}$ δQ $\frac{}{2}$

```
                        _____
                |     |
(enable)—        ████████
                |     |
                | REAL|
                |     |
(input parameter IN) —|IN  Q|— (output parameter Q)
                |_____|
```

注意

三角函数TRIG功能仅在 35x和 36x 系列 CPU, 9或更高版本r,和在 CPU352和CPU37x上的所有版本上有效.

参数

参数	说明
使能enable	当此功能被启动时，执行此项操作
IN	IN即被处理的.恒量或参考实数
ok	当此功能执行无溢出溢出时OK输出接通，除非一无效操作出现，和/或IN不是一个数（NaN.）
Q	输出即IN的三角函数值

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN								•	•	•	•	
ok	•											•
Q								•	•	•		

• 电流经功能模块时的有效参考（地址）或地点。

示例

在下例中， %R0001中值的COS被设置在 %R0033中

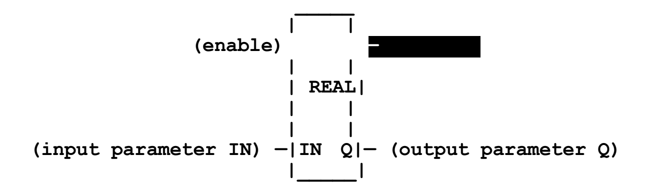
```
|             _____
|ALW_ON       |     |
|——] [———————| COS_|—
|            |     |
|            | REAL|
|            |     |
|    %R0001—|IN  Q|—%R0033
|            |     |
|   +3.141500|_____| -1.000000
|
```

对数/指数功能 (LOG, LN, EXP, EXPT)

LOG, LN, 和 EXP 功能有两个输入参数和两个输出参数。当此功能接收电流时，它执行输入IN中实数值的对数/指数函数操作，且将结果置与输出Q。

- 对于 LOG功能，IN的以10为底的对数值被设置在Q内。
- 对于 LN 功能，IN的自然对数被设置在Q内。
- 对于 EXP 功能函数，即IN的e的IN次幂，输出为Q。
  (注意: ***e*** is a constant used in logarithmic calculations. It has an approximate value of 2.71828.)
- 对于 EXPT功能，输入I1进行I2值的乘方运算，且结果被设置在输出Q内。(EXPT 功能有三个输入参数和两个输出参数)。

OK输出将接收到电流，除非IN为NaN (不是一个数) 或是负数。



参数

参数	说明
使能enable	当此功能被启动时，执行此项操作。
IN	IN 即运算的实数值。
ok	当此功能被执行无溢出时，OK输出接通，除非一无效操作出现，和/或IN为NAN或是负值
Q	输出Q包括IN的对数/指数值。

注意

LOG, LN, EXP和 EXPT 功能仅使用于 35x和 36x 系列的 CPU, 9.00或更高的版本，和 CPU352和 CPU37x.的所有版本。

注意

对于EXP功能，当输入值IN，是负无穷大时，按预期这个功能返回0的值。 既然如此，对于CPU352,这个功能不能通流，对于所有其他 90-30 CPU, 这个功能不能通流, 甚至输出为 0. (数值除0将溢出.它将出项在编程器窗口作为OVERFLOW.)

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN*								•	•	•	•	
ok	•											•
Q								•	•	•		
I1*								•	•	•	•	
I2*								•	•	•	•	

* 对于 EXPT 功能，输入IN被参数I1和I2代替。

• 电流经功能模块时的有效参考（地址）或地点。

示例

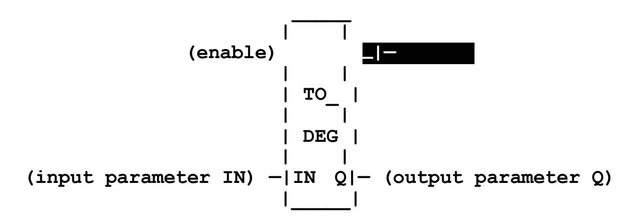
在下例中， %AI0001中的值是+3.000000,进行 +2.500000次乘方,结果为t, +15.58846, 并被放在地址 %R0001.中。

```
|              _____
|ALW_ON       |     |
|——] [———————|EXPT_|—
|             |     |
|             | REAL|
|             |     |
|             |     |
|   %AI0001—|I1   Q|—%R0001
|  +3.000000 |     | +15.58846
|             |     |
|             |     |
|     CONST —|I2    |
|   +2.500000 |_____|
|
```

弧度转换 (RAD, DEG)

当此功能接收电流时，相应的转换(RAD_TO_DEG或 DEG_TO_RAD，即：弧度至度，反之亦然)在输入IN内实数被执行，结果置于Q内。

OK输出将接收到电流，除非IN为NAN(不是一个数)。



参数

参数	说明
enable	使能enable 当此功能被启动时，执行此项操作。
IN	IN 即运算的实数值。
ok	当此功能被执行无溢出时，OK输出接通，除非IN为NAN
Q	输出Q即被转换的IN值。.

注意

弧度转换功能仅仅使用于 35x和 36x 系列的 CPU, 9.00或更高的版本，和 CPU352和 CPU37x.的所有版本。.

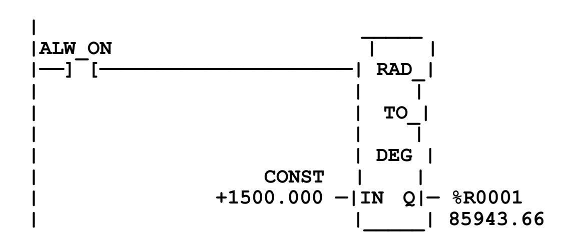
有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN								•	•	•	•	
ok	•											•
Q								•	•	•		

• 电流经功能模块时的有效参考（地址）或地点。

示例

在下例中，+1500被转换成DEG并被设置在%R0001地址内。.



Chapter 7

第七章　关系运算功能模块

关系运算功能模块是用来比较两个数值，这章将介绍下列关系运算功能模块：

缩写	功能	说明	Page
EQ	相等	检测两个数相等	7-2
NE	不等于	检测两个数不相等.	7-2
GT	大于	检测一个数大于另一个数	7-2
GE	大于或等于	检测一个数大于或等于另一个数	7-2
LT	小于	检测一个数小于另一个数	7-2
LE	小于或等于	检测一个数小于或等于另一个数.	7-2
RANGE	区间	判定一个数是否属于某个区间（4.5或更高版本的CPU）.	7-4

Standard 关系运算功能模块 (EQ, NE, GT, GE, LT, LE)

当这一功能模块就收电流通时，它便比较输入的两个参数I1 和 I2, 这些参数必须是相同的数据类型。关系运算模块按下列数据类型进行运算：

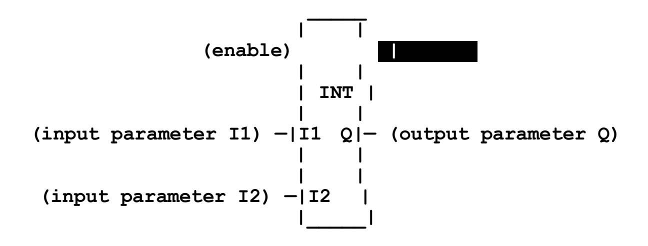
数据类型	说明
INT	带符号整型
DINT	双精度带符号整数
REAL	浮点数（对于RANGE功能无效）

注意

浮点数据类型仅使用于 35x和 36x 系列的 CPU, 9.00或更高的版本，和 CPU352和 CPU37x.的所有版本。当关系运算使用REAL型数据类型执行成功时，%S0020系统位置为ON。特别注意的是当另一个输入为NaN (不是一个数)时，关系运算功能模块不接受REAL数据类型 。

缺省数据类型时带符号整型。为比较带符号整数或者双精度带符号整数或实数，在选定关系运算功能模块后，要选择新的数据类型。为比较其他类型数据或两个数据类型不同的数据，必须先用相应的变换功能模块（在第十一章“转换功能模块中介绍”），将这些数据变换为整数中的一种。

如果输入I1和I2与特定的关系运算相符，则输出Q将接通且置ON（1），否则，置OFF(0)..



参数

参数	说明n
enable	当功能允许操作（使能）时，则执行运算。
I1	I1 为待比较的第一个值为一常数或参考地址 (I1处于关系式左边，如 I1 < I2).
I2	I2 为待比较的第一个值为一常数或参考地址. (I2 处于关系式的右边，如 I1 < I2).
Q	当I1和I2慢足特定关系时，输出Q接通。.

注意

I1和 I2必须为有效数，即不能为NAN（非数值）

Expanded Description

功能	说明n
等于	等使能通流，如果输入I1等于输入I2，则输出Q导通。
不等于	等使能通流，如果输入I1不等于输入I2，则输出Q导通。
大于	等使能通流，如果输入I1大等于输入I2，则输出Q导通。
大等于	等使能通流，如果输入I1大于或等于输入I2，则输出Q导通。
小于	. 等使能通流，如果输入I1小于输入I2，则输出Q导通。
小于等于l	等使能通流，如果输入I1小于等于输入I2，则输出Q导通。.

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
I1		o	o	o	o		o	•	•	•	•†	
I2		o	o	o	o		o	•	•	•	•†	
Q	•											•

• 电流经功能模块时有效参考地址或地点

. o 仅用于INT数据的有效参考地址，对DINT或REAL型则无效.

† 用 Logicmaster PLC 软件编程时对于双精度有符号整型运算常数限制为整型数值范围(+32,767 ～ –32,768) . 使用 VersaPro 软件编程时，允许双精度有符号整型范围.

示例

在下例中，两个双精度带符号整数， %R00100/101 和 %R00102/103，只要%I0001置为ON时，便进行比较，.如果输入值I1小于等于I2，线圈%Q00002将被接通，在下例中因为I1大于I2，所以线圈%Q00002状态为OFF.

```
|              _____
|%I0001       |     |
|--| |--------| LE_ |
|             |     |
|             | DINT|
|     %R00100 |     |                                                    %Q00002
|   +0000134689-|I1  Q|---------------------------------------------------( )-
|             |     |
|    %R00102  |     |
|  +0000134600-|I2   |
|             |_____|
|
```

区间 (INT, DINT, WORD)

区间（ RANGE ）功能模块用于确定一个值是否在两个数的范围之间。

注意

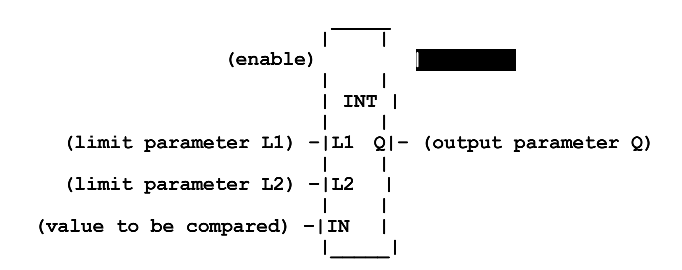
此功能仅适用于4.41 或更高版本的CPU

RANGE功能依靠以下数据类型操作 (REAL数据类型不适用于 RANGE功能模块):

数据类型	说明
INT	带符号整数
DINT	双精度带符号整数
WORD	字数据类型

缺省数据类型为INT型，不过，在选择此功能之后，它可以改变。要获得关于数据类型更多的信息，请参考第二章第二节“程序组织和用户参考/数据”。

当此功能被启动时，RANGE功能模块将比较输入参数IN内的值，并与规定的材树L1和L2所参考的范围相对照。当这个值在L1和L2确定的范围内时，包括L1和L2在内，输出参数Q被设定为ON（1），否则，Q被设定为OFF（0）。



注意

限制参数L1和L2表示一个范围的端点这两者中的任一参数设有被指定最大/最小或高/低的含义。因此，从0～100的所需要的区间可通过注定0至L1，100至L2，或0至L2，100至L1而确定。

参数

参数	说明
enable	当此功能被启动时，执行此操作.
L1	L1即此范围的起始点.
L2	L2即此范围的末点.
IN	IN即与L1和L2.所参考区间相对照的一个输入。
Q	当IN内的值在L1和L2参考的区间内，包括L1，L2在内时，输出Q接通。

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
L1		o	o	o	o		o	•	•	•	•‡	
L2		o	o	o	o		o	•	•	•	•‡	
IN		o	o	o	o		o	•	•	•		
Q	•											•

• 电流经功能模块时的有效参考（地址）或地点。

o 仅用INT或 WORD数据的有效参数，对DINT则无效。

‡ 对于DINT操作，恒量被限制在整数值

示例 1

在下例中%AI0001被检验在2个常数，0～100所参考的区间之间。

```
|
|%I0001   _____
|--| |----|RANGE|
|         | INT |
|         |     |                                          %Q0002
|   100 --|L1  Q|------------------------------------------( )-
|         |     |
|     0 --|L2   |
|         |     |
|%AI0001--|IN   |
|         |_____|
|
```

RANGE对应示例1的真实表				
启动状态 %I0001	L1 值常量	L2值常量	IN 值 %AI0001	Q 状态 %Q0001
ON	100	0	< 0	OFF
ON	100	0	0 — 100	ON
ON	100	0	> 100	OFF
OFF	100	0	Any value	OFF

示例 2

在下例中%AI0001 被检验在两个寄存器值所参考的范围之间。

```
|
|%I0001   ______
|--| |----+RANGE|                                                    %Q0002
|         | INT |
|         |     |
|%R0001 --+L1  Q|----------------------------------------------------( )-
|         |     |
|%R0002 --+L2   |
|         |     |
|%AI0001--+IN   |
|         |_____|
|
```

RANGE对应示例2的真实表				
启动状态 **%I0001**	**L1 值 %R0001**	**L2 值 %R0002**	**IN 值 %AI0001**	**Q 状态 %Q0001**
ON	500	0	<0	**OFF**
ON	500	0	0-500	**ON**
ON	500	0	>500	**OFF**
OFF	500	0	any value	**OFF**

Chapter 8

第八章　位操作功能模块

位操作运算功能模块将按位串进行逻辑比较和移位操作。按单字的AND, OR, XOR, 和 NOT功能操作。保留位操作可按多字进行，最大位串长度是256个字。位操作功能需要字数据。

尽管数据必须定为以16位递增，但这些功能 仍将按作为连续位串的数据进行操作运算。用第一个字的位作为最低的有效位（LSB），最后一个字的最末位为最高有效位（MSB），例如，如果你从地址%R0100开始，规定数据的三个字，那么，将在48个连续位上进行操作。

%R0100	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**ϒ⌋bit 1 (LSB)**
%R0101	**32**	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	
%R0102	**48**	**47**	**46**	**45**	**44**	**43**	**42**	**41**	**40**	**39**	**38**	**37**	**36**	**35**	**34**	**33**	
	ϒ⌉ (MSB)																

注意

在多字运算中，重叠输入和输出的参考地址范围，会产生预想不到的结果。

下面是这章内所介绍的位操作功能模块：

缩写	功能	说明	
AND	逻辑与	如果位串L1中的一个位与位串L2中的相对应的位都是1，那么设置1在输出位串Q中的相应的位置中。	8-3
OR	逻辑或	如果位串L1中的一个位与/或位串L2中的相对应的位都是1，那么设置1在输出位串Q中的相应的位置中	8-3
XOR	逻辑异或	如果位串L1中的一个位与/或位串L2中的相对应的位不同，那么设置1在输出位串Q中的相应的位置中	8-5
NOT	逻辑非	设定输出位串Q中每个位的状态至位串I1中相应位的相应状态。	8-7
SHL	左移	沿被给定数器的位置将一个字或一串字中的所有位移至左边	8-8
SHR	右移	沿被给定数器的位置将一个字或一串字中的所有位移至右边	8-8
ROL	左循环	沿给定数器的位置，将一串中的所有位循环至左边。	8-10
ROR	右循环	沿给定数器的位置，将一串中的所有位循环至右边。	8-57
BTST	位测定	测定位串中的一个位目前是否位1或0。	8-12
BSET	位设定	设定位串中的一个位为1	
BCLR	位清除	用设定这个位为0可清除一位串中的位.	8-14
BPOS	位定位	定位一位串中的一个位定到1	
MSKCMP	屏蔽比较	比较两个带有屏蔽选择位的能力独立位串的内容（4.5或更高版本CPU有效).	8-18

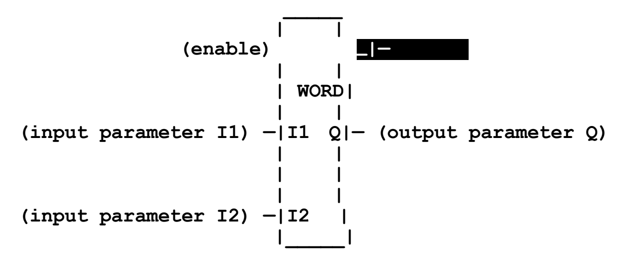
AND 和 OR (WORD)

电源接通的每次扫描，ADD或OR功能模块将检验从每个数据最低有效位开始的位串L1中的每一位及位串I2中的相应位。
对于被AND功能所检验的每两位，若都为1，则在输出位串Q的相应存贮单元置如1，如果其中一位或两位都位0，则该存贮单元中输出位串Q置为0。
AND功能用于这只掩蔽或屏蔽是有效的。当其仅有某位通过（这与屏蔽中的1相反），而所有其他位均置0。AND功能也可清除所选的字存贮区，这则通过与另一个已知的保持全0的位串逐位逻辑与而达到的。所指定的位串I1与I2可重叠。

对于OR功能所检验的每两个位，如果其中一位或两位均为1，则1置入输出位串Q的相应存贮单元；如果两个位均为0，则0置入位串Q中的相应存贮单元。

OR功能用于综合位串是有效的。可通过用一简单的逻辑装置来控制许多个输出。这一功能相当于通过位串中位数相乘的两个继电器的并联触点。它可用于驱动直接辨别输入状态的指示灯，或在状态指示灯上叠加闪光条件。

通电时，功能模块即向右输送电流。



参数

参数	说明
使能enable	当功能块被使能时，执行此功能。
I1	I1 即常量或第一个字符串首个字的参考地址。
I2	I2即常量或第一个字符串首个字的参考地址。
ok	只要有使能产生，OK即有输出。
Q	输出 Q 即操作的结果。

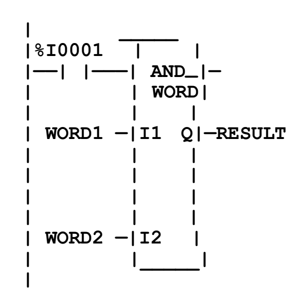
有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
I1		•	•	•	•	•	•	•	•	•	•	
I2		•	•	•	•	•	•	•	•	•	•	
ok	•											•
Q		•	•	•	•	•†	•	•	•	•		

• 当电流通过功能时的有效参考地址或位置n.
† 仅适用%SA, %SB, 或 %SC; 不可用%S.

示例

在下例中，只要%I0001被置位，别名位WORD1和WORD2的两个16位字符串将进行检验，AND逻辑运算的结果存在输出RESULT里。



WORD1 (I1)	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**0**
WORD2 (I2)	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**

RESULT (Q)	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**

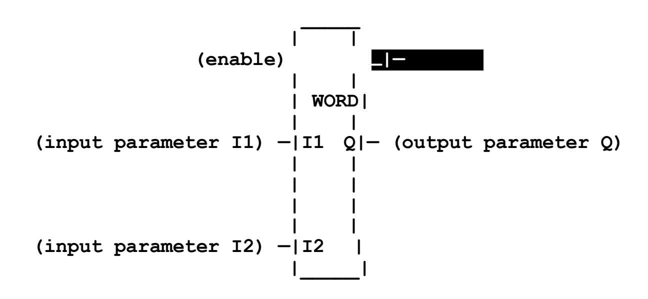
XOR (WORD)

异或（XOR）运算功能被用来将输入I1字符串中的每一位与输入I2字符串中的相应的位进行比较，如果相应的位不同，将在输出字符串的相应位置置1。

对快速比较两个字符串，或每两次扫描一个ON状态来闪烁一组位，XOR功能是有效的。

每次接受能量的扫描，功能模块将从每个字符串的最低有效位开始，检验字符串I1中的每一位及字符串I2中的相应的位，对被检验的两个位来说，如果仅一位是1，那么1便置入字符串Q中的相应的位，当接收电流时，XOR功能模块接通电源将向右传递电流。

如果字符串I2和输出字符串Q从相同的地址开始，那么，置入字符串I1的一个1会引起I2相应位在0和1之间的交替变化，只要是接收能量，便随着每次扫描而改变状态。通过以两倍所期望的速率对功能模块脉冲式输能，可编程较长的周期，输能脉冲是一次扫描的长度（单稳态线圈或自复位定时器）



参数

参数	说明
enable	当功能块被使能时，执行此功能。
I1	I1即常量或被XOR的第一个字符串首个字的参考地址。
I2	I2即常量或被XOR的第二个字符串首个字的参考地址。.
ok	只要有使能产生，OK即有输出。
Q	输出 Q I1与I2执行 XOR操作的结果。

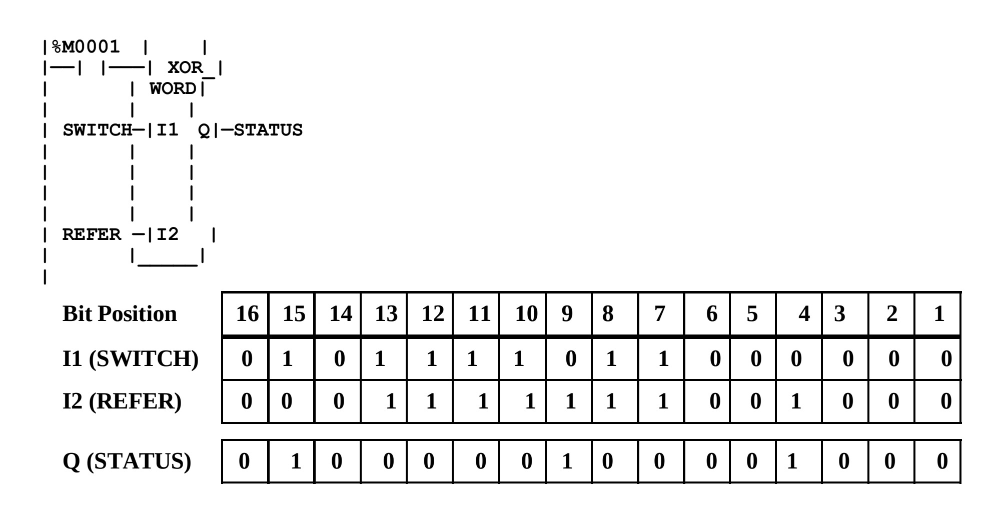
有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	cons	none
enable	•											
I1		•	•	•	•	•	•	•	•	•	•	
I2		•	•	•	•	•	•	•	•	•	•	
ok	•											•
Q		•	•	•	•	•†	•	•	•	•		

• 电流经功能模块时的有效参考地址或位置
† %SA, %SB,或%SC可用, %S不可用。

使用XOR告警电路示例

本例中，使能%M0001为ON, 16位字符串SWITCH和位串REFER比较。SWITCH 表示报警on/off状态. REFER位串代表正常或非报警状态. 如果 SWITCH 某位状态和相应的REFER位状态不同，对应的输出Q为1. 在正常情况(没有报警)下，STATUS数值为0.



Bit Position	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
I1 (SWITCH)	0	1	0	1	1	1	1	0	1	1	0	0	0	0	0	0
I2 (REFER)	0	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0
Q (STATUS)	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0

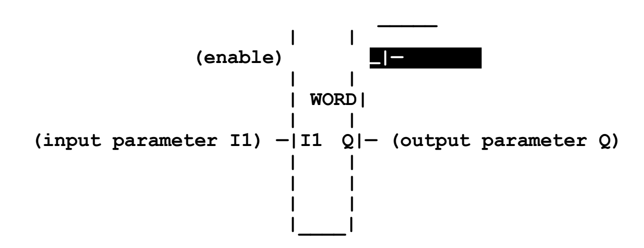
STATUS数值可用于NE功能一个输入,对比STATUS和常数0. 如果STATUS不等于0, 输出为ON,表示有故障.

可使用BPOS功能查找STATUS中等于1的位, BPOS位查找STATUS位并且报告第一个为1的位的位置 (1 ～16).上例中, BPOS输出数值4,表示第四位为1. 为了检测更多位，你可以存储第四位,使用 BCLR功能清除第四位,重复BPOS命令查找下一个等于1的位 (上例中是第九位). 重复该过程直到查找出所有非零的位. 注意BCLR和BPOS功能详细信息在本章讨论.

NOT (WORD)

求非功能用来将输出字符串Q的一位设置成与字符串I1相应位相反的状态。

如果求非功能的使能（enable）被接通，在每次扫描周期中将执行，并将流通电流。



参数

参数	说明
enable	当 功 能 块 的 使 能 被 接 通 时 ， 此 功 能 将 执 行 。
I1	I1 即常量或求反操作的字的参考地址.
ok	只要有使能产生，OK即有输出。.
Q	输出Q即I1的NOT（求反）

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
I1		•	•	•	•	•	•	•	•	•	•	
ok	•											•
Q		•	•	•	•	•†	•	•	•	•		

• 电流经功能模块时的有效参考地址或地点

† %SA, %SB,或%SC可用, %S不可用。

示例

在下例中，只要%I0001被置位，别名为NOTCAT的位字符串将被置为字符串CAT的相反字符串。真实值看下表中

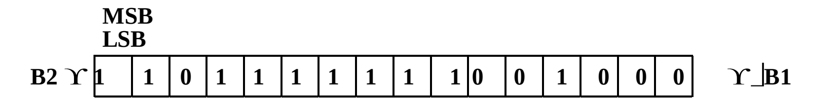
```
|
|%I0001   _____
|——| |———| NOT_|—
|         | WORD|
|         |     |
|   CAT —|I1  Q|—NOTCAT
|         |     |
|         |     |
|         |     |
|         |_____|
|
```

CAT	1	1	0	1	0	1	0	0	1	1	0	1	0	0	0	1
NOTCAT	0	0	1	0	1	0	1	1	0	0	1	0	1	1	1	0

SHL 和 SHR (WORD)

左移功能（SHL）用来将一个字或一字组的左移指定的位数，当发生左移时，输出字符串将向左移出指定的位数，随着字符串高末位的左移出，相同的数将从字符串的低末位移入。

MSB
LSB



右移功能（SHR）用来将一个字或一字组的右移指定的位数，当发生右移时，输出字符串将向右移出指定的位数，随着字符串低末位的右移出，相同的数将从字符串的高末位移入。

MSB
LSB

B1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | B2

这两种功能可选择字符串的长度为1～256个字。

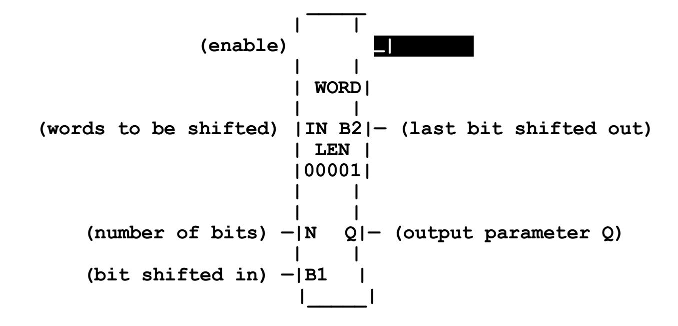
如果移位的位数（N）大于数组（LEN）*16的位数，或如果移位的位数为零，那么数组Q用输入位（B1）备份填满，且输入位还复制到输出电流（B2）中。如果一位的位数时零，那么无移位过程，输入组复制到输出组中，输入位（B1）复制到电流内。

开始移入位串起始点的位规定要通过输入参考数（B1）。如果指定的所移位数的长度大于1，即位串中的每一位均填满相同的值（0或1），它可以是：

- 永远接通触点(%S7)，保持B1永远为1。
- 永远关断触点(%S8)，保持B1永远为0。
- 触点来自内部线圈例如%M或%Q允许你改变数值。
- %I 触点允许你改变输入触点数值。

除非指定移位的位数为零，SHL和SHR功能将向右传递电流。

移位后的输出Q是输入字符串的复制，如果你想输入的字符串被移位，那么输出参数Q必须与输入参数IN使用相同的本地存贮器，SHL和SHR功能模块在使能为ON的每个扫描周器都将执行，输出B2保持着最后移出的一位，例如，如果移4位，B2将位第四位（1或0）



参数

Parameter	Description
enable	当使能为1是，移位功能块将被执行.
IN	IN即移动的第一个字.
N	N 即数组位移的地址 (位) 数
B1	B1 位要移入数组的位的值
B2	B2 为要移出数组的最后位的值
Q	Q 为移位数组的第一个数
LEN	LEN为数组中要移位的字数.

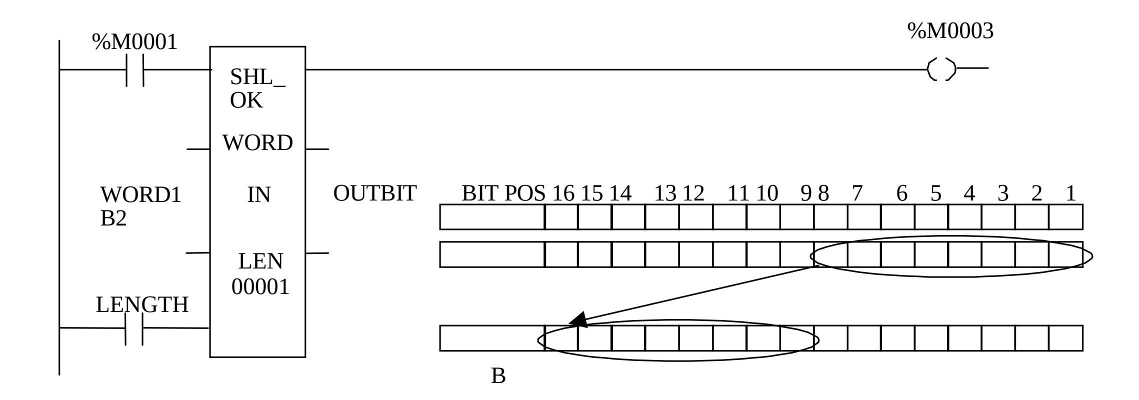
有效存贮器类型

Parameter	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
I		•	•	•	•	•	•	•	•	•		
N		•	•	•	•		•	•	•	•		•
B1	•											
B2	•											•
Q		•	•	•	•	•†	•	•	•	•		

• 电流经功能模块时的有效参考地址或地点
† %SA, %SB, or %SC only; %S cannot be used.
%SA, %SB,或%SC可用, %S不可用。

示例

在下例中，当输入%M0001为ON时，SHL功能将做IN(别名 WORD1)的一个复制，然后，在这个复制中，它将所有的位左移8位(N移动位数)，位9－16被移出（丢弃）， 1－8位占用9－16位，当1－8位移动时，原1－8位将“腾空”
在这个例子中，当触点%M0002关闭时，输入B1等于1，最后移动和填充的字将被写入到输出Q（别名为WORD2）的地址中。 原来IN处的WORD1的值没有改变，输出B2等于0因为最后位被移出为0（被9位占用），并且线圈%M0003为ON，因为功能正确工作并且在输出OK处有电流产生。
在下例中，当输入%M0001为ON时，SHL功能将做IN的一个复制。



ROL 和 ROR (WORD)

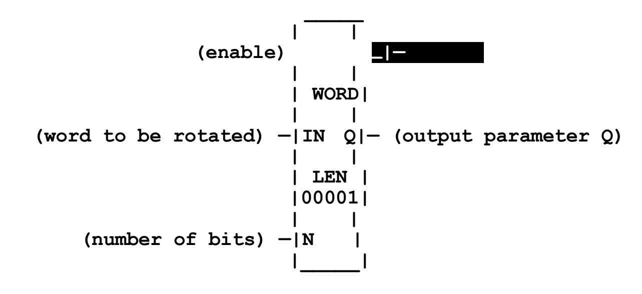
左循环功能（ROL）用于在一个指定地址的一个字符串向左循环所有位。当循环开始时，所指定的位数向左循环移出输入字符串，再进入右边的字符串。

右循环功能（ROR）用于在一个指定地址的一个字符串向右循环所有位。当循环开始时，所指定的位数向右循环移出输入字符串，再进入左边的字符串。

两种功能均可选字符串长度为1－256个字

指定的循环位数须大于0，而小于字符串中的位数。否则不移位。无电流产生。

循环后的结果放在输出字符串Q中，如果想输入字符串循环移动，那么输出参数Q必须与输入字符串占用相同的本地存贮器，在使能输入为ON时，在每个扫描周期功能将被执行。



参数

参数	说明
使能enable	当使能输入为ON时，功能被执行
IN	IN 被循环移动的第一个字的地址
N	N 数组循环移动的位数 (位)
ok	当功能模块被使能，并且循环移动位数大于0和不大于数组的位数时，OK产正输出。
Q	输出Q为循环数组的第一位
LEN	LEN 循环数组的长度 (1 – 256)

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN		•	•	•	•	•	•	•	•	•		
N		•	•	•	•		•	•	•	•	•	
ok	•											•
Q		•	•	•	•	•†	•	•	•	•		

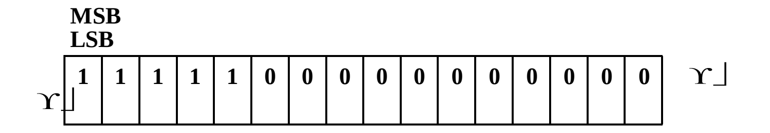
• 电流经功能模块时的有效参考地址或地点.
† %SA, %SB, or %SC only; %S cannot be used.
%SA, %SB,或%SC可用, %S不可用。

示例

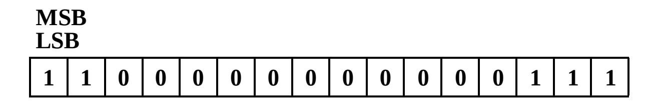
在下例中，只要输入%I0001为ON,ROL将对输入字符串IN做复制，于是，在复制中，将输入字符串循环3位(N中的数值)并结果放在%R0002中。执行功能后，输入字符串%R0001并没有改变，如果想改变输入字符串，那么在IN和Q中使用相同的参考地址。

```
|
|%I0001  _____
|--| |---|     |
|        | ROL_|-
|        | WORD|
|        |     |
| %R0001-|IN  Q|-%R0002
|        |     |
|        | LEN |
|        |00001|
|        |     |
| CONST -|N    |
| +00003 |_____|
|
```

%R0001:



%R0002 (**%R0001**置位后):

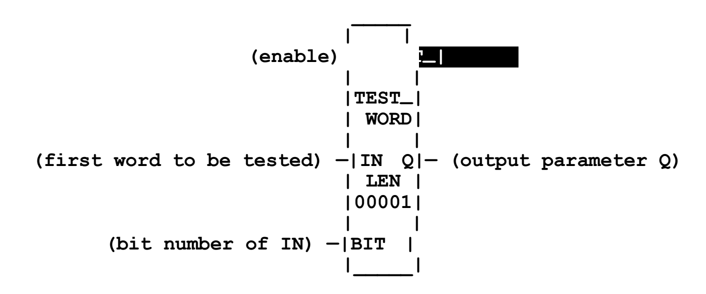


BTST (WORD)

位检测功能(BTST)用于检测一个字符串中的一位，来判定那一位是1还是0，检测的结果置于输出Q中。

每次扫描到电源接通时，BTST功能模块设置与指定位相同的状态到输出Q中。如果一个寄存器而非一个常数用作指定位数。则同样的功能模块可按逐位扫描的不同位进行检测。如果BIT值超出范围（1≤BIT≤（16 * LEN））时，则Q置为OFF.

一个1～256字长度的字符串可供选择。



参数

参数	**说明**
使能enable	当功能模块被允许时，则执行位检测。.
IN	IN 运算数据的第一个字。
BIT .	BIT 需检测的IN的位数，有效范围是（1≤BIT≤（16 * LEN））
Q	Q 如果位检测是1，则输出Q.
LEN	LEN检测字符串的字数。

注意

当使用 Bit Test, Bit Set, Bit Clear 或 Bit 功能模块时, 位数是1－16，而不是0－15。.

有效存贮器类型

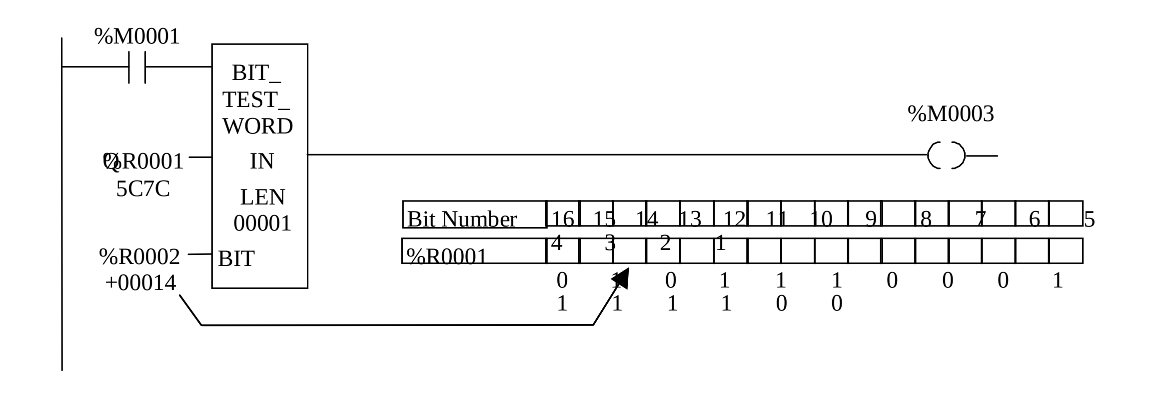
参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN		•	•	•	•	•	•	•	•	•		
BIT		•	•	•	•		•	•	•	•	•	
Q	•											•

• 电流经功能模块时的有效参考地址或地点.

示例

在下例中，只要输入%M0001为ON，在字%R0001中的14位被检测。

(%R0002中设定的数值，即14位). 因为在 %R0001中14位为0(5C7C), 则输出 Q 不能转换为ON, 注意，功能块仅是WORD类型，因此，在IN中使用的存贮器地址在编程器窗口中以16进制的格式显示，然而，不管是否是使用常数或存贮器地址，BIT的值都将以整数形式出现。

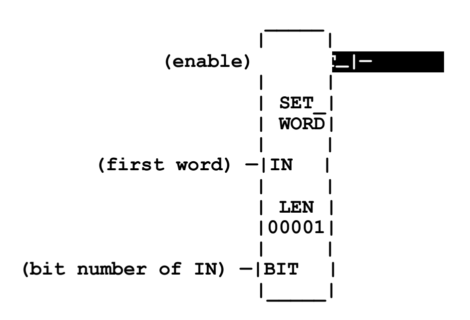


BSET 和 BCLR (WORD)

位置位(BSET)功能用于将字符串中的一位设置为1，位清除(BCLR)功能用来将字符串所包含的一位设置为0。

每次扫描到电源接通是，功能模块BSET将指定的为设置为1或BCLR将指定的位设置为0。如果一个变量（寄存器）而不是一个常数被用来做指定的位数，则功能模块在连续扫描中将设定不同的位。

可选字符串的长度为1～256，功能模块将向右传递电流，除非BIT的数值超出范围（1≤BIT≤（16 * LEN）），OK置为OFF。例如，如果LEN设为1，则字符串的位长度被检测出为16，在这中情况下，BIT的数值设为17或更高，它将超出范围，OK输出将不能置位ON.



参数

参数	说明
使能enable	当使能输入为ON时，位操作功能将被执行。
IN	IN被执行操作数据的第一个字
BIT	BIT 在IN中被置位或清除的位数，有效范围是（1≤BIT≤（16 * LEN））
ok	只要使能产生，BIT数值在有效范围之内，OK就有输出。T .
LEN	LEN在IN配置起始地址的字符串的字数..

注意

当使用 Bit Test, Bit Set, Bit Clear 或 Bit 功能模块时，位数是1－16，而不是0－15。

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN		•	•	•	•		•	•	•	•		
BIT		•	•	•	•		•	•	•	•		•
ok	•											•

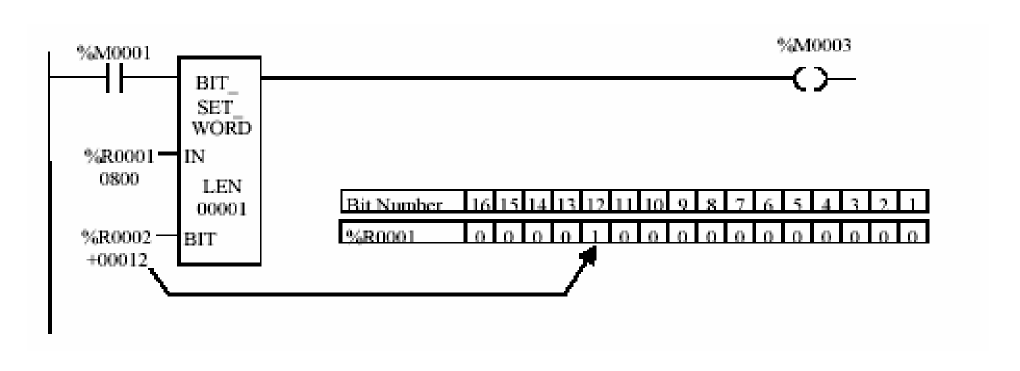
• 电流经功能模块时的有效参考地址或地点.

† %SA, %SB,或%SC可用, %S不可用

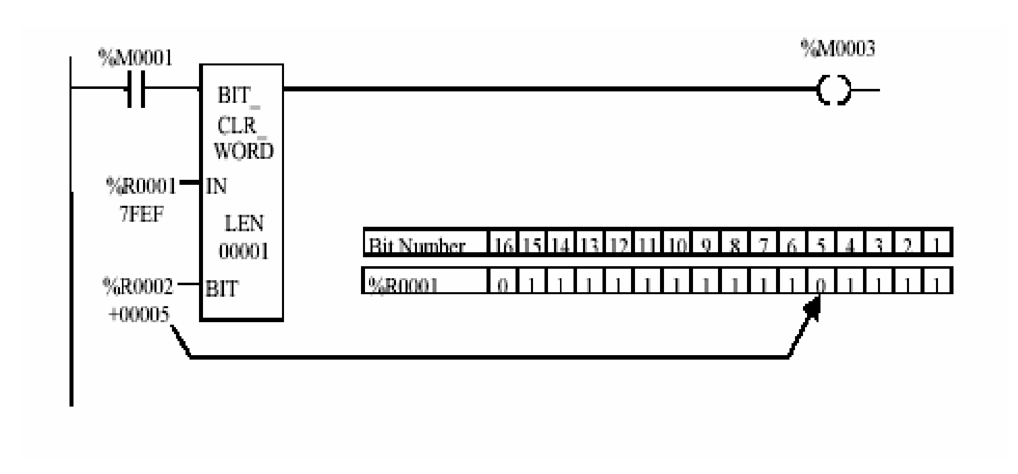
示例

注意，位置位和位清除功能块仅是WORD类型，因此，在IN中使用的存贮器地址在编程器窗口中以16进制的格式显示，然而，不管是否是使用常数或存贮器地址，BIT的值都将以整数形式出现

在下例中，当输入%M0001为on,在指定地址%R0001（IN中的地址）字符串的第12(BIT中输入的位数) 被设置为1 (置位).



在下一个例子中，当当输入%M0001为on,在指定地址%R0001（IN中的地址）字符串的第5位(BIT中输入的位数) 被设置为0 (清除)。

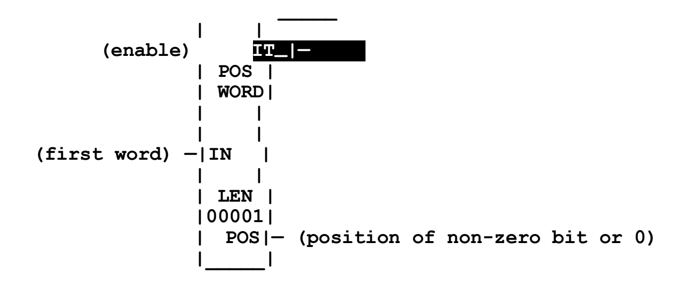


BPOS (WORD)

定位（BPOS）功能模块用在本地的字符串中确定数值为1的位置。在每次扫描中功能模块将被执行，扫描IN中的字符串，直到发现字符串中为1的位，或整个字符串被扫描完成时，功能模块才停止扫描。

POS功能查找到位串中第一个非零位时输出该位位置; 如果未找到非零位输出0.
POS在字符串内第一个非零处理位，如未到非零位，则POS置0。

可选字符串的长度为1～256，只要使能为ON,功能模块便向右传递电流。



参数

参数	说明
enable	当使能输入为ON时，位操作功能将被执行。.
IN	IN被执行操作数据的第一个字
ok	如果有使能时，OK便产生输出。.
POS	非零位的位置，或非零位没有发现。
LEN	LEN 字符串的长度.

注意

当使用 Bit Test, Bit Set, Bit Clear 或 Bit 功能模块时, 位数是1－16，而不是0－15。

有效存贮器类型

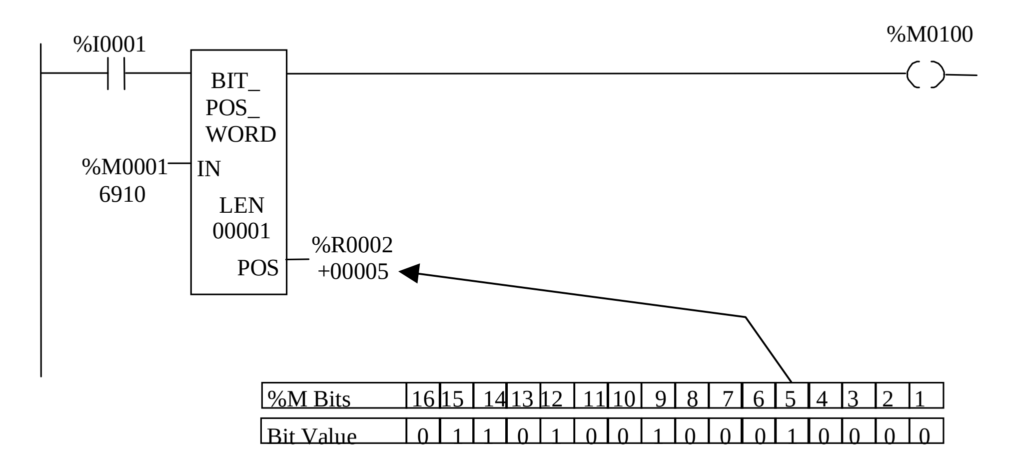
参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN		•	•	•	•	•	•	•	•	•		
POS		•	•	•	•		•	•	•	•		
ok	•											•

• Valid reference or place where power may flow through the function.

示例

注意，位置位和位清除功能块仅是WORD类型，因此，在IN中使用的存贮器地址在编程器窗口中以16进制的格式显示，然而，POS中的值将以整数形式显示，编程器以16进制的格式显示IN中的前16位。

在下例中，如果%I0001为ON，字符串%M0001将被查找直到发现等于1的位，或整个字符串被查找完，线圈才被置为ON。如果发现等于1的位，它在字符串中的位置写入到%R0002中，否则将0写入到%R0002中。示例中显示，在查找中在第五位中第一次遇到1，所以%R0002写入的值是5。



MSKCMP (WORD, DWORD)

屏蔽比较(MSKCMP)功能模块(*对于 4.41或更高版本的 CPU有效*)用来比较带有屏蔽选出能力的两个独立的字符串的内容，被比较字符串的长度由LEN参数（对于MSKCMPW功能，LEN的值参考16位字的数目，对于MSKCMPE功能参考32位双字的数目）

当功能模块的使能为ON时，功能模块从第一个字符串与第二个字符串中相应的位比较，直到miscompare被发现，或比较完成。这个功能在使能为ON时每次扫描都执行，所以多于多种用途，脉冲信号常作为使能输入。

BIT输入用于存储下一次比较开始的位的位置(*0*表示位串第一位). BN输出用于存储上一次结束比较的位的位置(*1*表示位串第一位). BIT和BN使用同样的参考地址会导致下一次比较从上一次结束比较的那位的下一位开始，或者如果所有位都相同那么下一次比较从第一位开始。

如果你希望从位串中其他位置开始下一次比较，BIT和BN应设定不同的地址，如果BIT的值大于位串长度，开始下一次比较之前BIT被复位为0。

如果I1和I2所有位相同

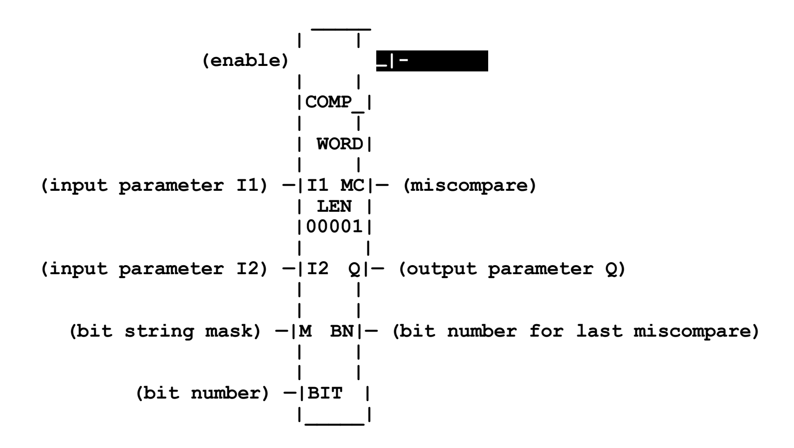
如果I1和I2所有对应的位都相同，那么输出MC为0且BN为输入位串的最高位，比较结束。下一次比较启动时，BN被复位为0。

如果两个位串不一样

如果比较的两个位串不一样，检验M中相对应的位。如果屏蔽位为*1*, 继续比较直至达到另一个不同位或直至位串末尾。

如果检测到不同的位，但相应的屏蔽位为0，那么执行以下几点：

1. 设定M相对应的位为1.
2. 设定输出MC为1.
3. 更新输出Q与M相等.
4. 设定BN为检测到不同的位的位置号.
5. 结束比较.



参数

参数	说明
enable	使能为ON,执行该功能
I1	第一个被比较的变量
I2	第二个被比较的变量
M	屏蔽字（比较的两个位不相等时，对应的位置1）
BIT	下一次比较开始的位的位置
MC	发现比较的两位不相等时置 1
Q	与M相等
BN	上一次结束比较的位的位置
LEN	被比较变量长度

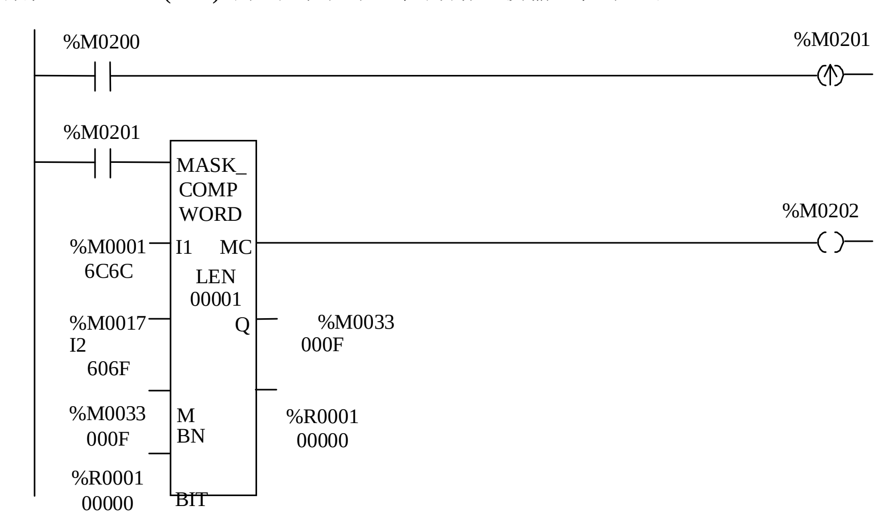
有效存储器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
I1		o	o	o	o	o	o	•	•	•		
I2		o	o	o	o	o	o	•	•	•		
M		o	o	o	o	o†	o	•	•	•		
BIT		•	•	•	•	•	•	•	•	•	•	
LEN											•‡	
MC	•											•
Q		o	o	o	o	o†	o	•	•	•		
BN		•	•	•	•	•	•	•	•	•		

• 参考地址或位置电流流过功能块.
o 对于WORD有效；对于DWORD无效.
† 仅支持%SA, %SB, %SC; %S不能使用.
‡ 对于WORD最大常数值4095对于DWORD为2047 .

示例 1 – MSKCMP 指令

当%M0200 得电，触点%M0201跳变线圈得电一个周期，使得MSKCMP功能执行一次. %M0001 到 %M0016 (I1) 和 %M0017到 %M0032 (I2)对比.%M0033到%M0048 (M) 包含屏蔽值. %R0001 (BIT) 为0表示从第一位开始比较输入位串I1和I2。



第一次执行该功能之前，变量状态如下：

执行MSKCMP之前输入变量内容:

%M 位	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
输入1(I1)	0	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0

%M 位	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
输入2(I2)	0	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1

%M 位	48	47	46	45	44	43	42	41	40	39	38	37	36	35	35	33
屏蔽 (M/Q)	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

BIT/BN (%R0001) = 0
MC (%M0202) = OFF

第一次执行MSKCMP以后变量状态如下：

下表说明执行MSKCMP一次以后Mask (M/Q)变量内容。(I1和I2 仍然保持上面数值) 第九位不相同, Mask位串第九位(%M0041)设为1, BIT/BN数值为9, 且MC输出为ON保持一个扫描周期。 尽管第一位和第二位不相等, MC不输出，因为这些位屏蔽位为1.

%M Bits	48	47	46	45	44	43	42	41	40	39	38	37	36	35	35	33
Mask (M/Q)	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1

BIT/BN (%R0001) = 9

MC (%M0202) = ON (for one scan)

示例2 – MSKCMP故障检测

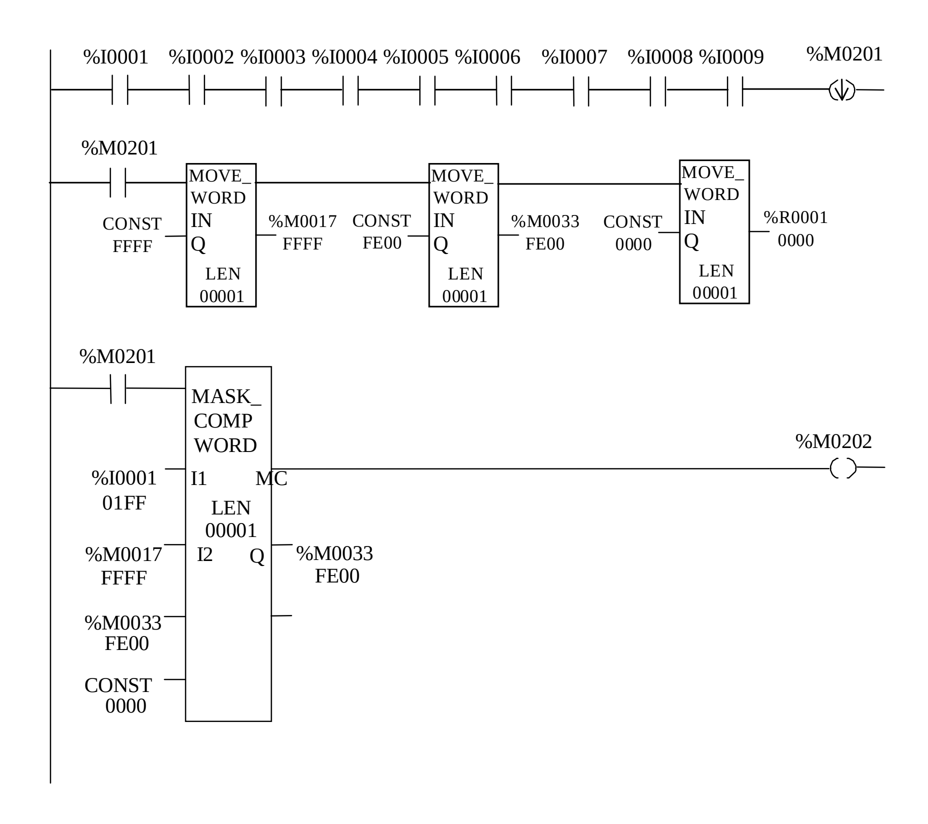
间歇性问题难于发现并处理。例如当几个接近开关用于串联电路，接通检测故障继电器。在正常情况下，所有开关闭合且检测故障继电器得电(自动防故障装置)。当故障发生时，其中一个触点打开且检测故障继电器失电。如果故障触点仍然打开，很容易发现那个接近开关引起故障。然而，有时触点仅打开很短时间，可能小于一秒，然后又关闭。这引起检测故障继电器断电很短时间且停止过程。既然触点又关闭，一切看上去正常。

为了帮助解决这样的问题，以上回路就像“故障捕捉器”因为它检测哪个触点打开并存储到寄存器中。第一行，接近开关触点，对应输入模块点 (%I1 – %I9)，串联接通%M0021,负跳变线圈.

第二行初始化 MSKCMP，用于捕获故障. 第一个Move指令向MSKCMP输入I2写1. 第二个Move写入1到屏蔽字位10—16 (因此这些位忽略),因此对于接近开关%I0001—%I0009仅比较前九位(MSKCMP 使用整个字). 第三个向输出寄存器移入0, %R0001，它用于报告最新故障.

正常运行时，接近开关都关闭，MSKCMP输入I1前九位为1. 输入I2初始化所有位为1表示正常状况下和输入接近开关对比. 因为只有九个接近开关所以mask位10—16 都设为1. 当一个接近开关打开, %M0201触点得电一个扫描周期. 第二行初始化执行，第三行MSKCMP使能. MSKCMP 对比输入接近开关和输入 I2 ,确认哪个接近开关为 0 (open)，记录打开接近开关号到BN (%R0001) . 从%I1开始一次编号1—9. 例如，如果 %I4 打开, %R0001 为数值 4.

注意，在这个回路中，如果接近开关打开后关闭，线圈 %M0201 断电后重新得电，但是打开的接近开关号将存储到%R0001中. 然而，如果接近开关又打开，例如，操作人员按下急停按钮或打开安全门，屏蔽比较再次激活并且记录最新打开的接近开关号到%R0001中. 这表示故障发生后设备不动直到%R0001数值被核对. 如果这样不可行，增加一个Move指令.


%I0001
%I0002
%I0003
%I0004
%I0005
%I0006
%I0007
%I0008
%I0009
%M0201
%M0201
MOVE_
WORD
IN
Q
LEN
00001
CONST
FFFF
%M0017
FFFF
CONST
FE00
%M0033
FE00
CONST
0000
%R0001
0000
MASK_
COMP
WORD
%I0001
01FF
I1
MC
%M0202
%M0017
FFFF
I2
Q
%M0033
FE00
CONST
0000

Chapter
9

第九章　数据传送功能模块

数据传送功能模块提供基本数据的传送能力，这章中介绍下例数据传送功能模块：

缩写	功能	说明	页数
MOVE	传送	以单个位进行数据复制，除MOVE_BIT是256位外，其他允许的最大长度是256个字，数据能构以不同类型进行传送，且实现不需要转换。	9－2
BLKMOV	块传送	把含有7个常量的数据块拷贝到指定地址中，这些常量是功能模块的一部分。	
BLKCLR	块清除	替换数据块中非零的数据为0，该功能可用于清除(%I, %Q, %M, %G, 或 %T)的位或字(%R, %AI, or%AQ) 中的存贮器。 最长允许256个字	
SHFR	移位寄存器	将一个或多个数据移入表中。最大允许256个字。	
BITSEQ	位定序器	通过一个位阵列完成一位的定序移动，最大长度为256个字。	
COMMREQ	通讯请求	允许程序通过智能模块通讯，像Genius通讯模块或可编程协处理器进行通讯。	

MOVE (BIT, INT, WORD, REAL)

MOVE功能模块用来将数据从一个存贮单元拷贝到另一个存贮单元,因为数据是以位格式拷贝的，所以新的存贮单元不需要与原存贮单元使用相同的数据类型。

MOVE功能模块有两个输入参数和两个输出参数，当功能模块使能时，它将以位的形式将数据从输入参数IN拷贝到输出参数Q中,如果数据是从一个类型的存贮单元到另一类型的存贮单元（如，从%I存贮单元到%T存贮单元），那么不管传送操作是否引起数字存贮单元改变状态，与数据存贮单元相联系的变换信息也会被刷新。输入参数的数据不会改变，除非在输入与输出中存在重叠的地址。

BIT类型需要注意的事项，如果Q参数上BITA数组不包含某字节中的所有位，那么该字节（不在数组内）相关的转换位在MOV-BIT导通时将被清除。

输入IN即可是传送数据的一参考地址，也可以是一常量，如果指定为一常量，那么，常量的值被放置到输出参考所指定的存贮单元。例如，对IN，如果一个常量的值被指定为4，那么4就被放置到由Q所指定的存贮单元内。如果长度大于1，而且一常量已被指定，那么常量被放置在由Q所指定的存贮单元和随后的单元，相当于指定长度。例如，如果一个常量值被IN指定为9，长度为4，那么9存入由输出Q所指定的存贮器存贮单元和后续的3个存贮单元。

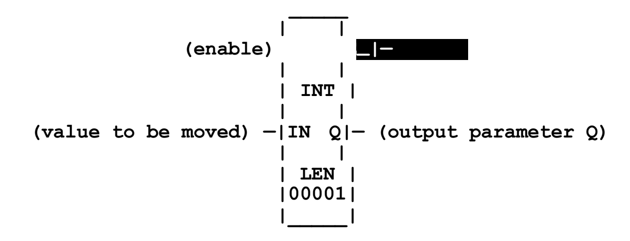
LEN操作数规定了以下数目：

- MOVE_INT和MOVE_WORD传送字.
- MOVE_BIT传送位.
- MOVE_REAL.传送实数

注意

REAL 数据类型仅使用于35x 和 36x 系列 CPU, 9 或更高版本, 和 CPU352与 37x的所有版本.

当功能块被使能时，即向由传递电流。



参数

参数	说明
使能enable	当.功能块被使能时，即执行传送。
IN	IN 即被传送的数据 (移动). 对于 MOVE_BIT, 所有参考地址都可以使用; 不需要数据排列. 然而, 从指定参考地址开始的16位数值在编程器窗口中显示。
ok	只要功能模块被使能时，OK就有输出。.
Q	当传送被执行时，IN的值拷贝到Q，对于MOV-BIT功能，所有的参考地址都可使用，不需要数据排列，不过从指定参考地址开始的16位数值在编程器窗口中显示。
LEN	LEN 即可以传送数据的位数，对于MOVE_WORD 和MOVE_INT, LEN 必须在1到256之间，对于 MOVE_BIT, 当IN是一个常量时，LEN必须在1到16位之间，否则LEN必须在1到256之间。

注意

对于351, 352, 36x 和 37x 系列 CPU, MOVE_INT 和 MOVE_WORD 功能不支持IN和Q参数的重叠, IN小于Q参考地址. 例如，以下的值：IN=%R0001, Q=%R0004, LEN=5 (字), %R0007和 %R0008 将是不确定的; 然而使用以下值: Q=%R0001, IN=%R0004, LEN=5 (字)有效.

同样,请注意仅仅是35x和36x系列CPU (9.00和更高版本), CPU35 和 37x 所有版本具有浮点功能，一次仅仅90-30系列CPU可以使用MOVE_REAL功能。

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN		•	•	•	•	o	•	•	•	•	•	
ok	•											•
Q		•	•	•	•	o†	•	•	•	•		

注意: 对于REAL型数据, 有效的类型是 %R, %AI, 和 %AQ.

• 当电流通过功能块时，对于BIT, INT, 或WORD数据类型的有效参考地址,或位置，对于MOVE_BIT, 可以使用离散的参考地址%I, %Q, %M, 和 %T，不需要排序

o 仅仅对于 BIT或 WORD型数据有效的参考地址，对 INT无效.

† 仅仅可使用%SA, %SB, %SC; %S不可用

示例1 - 重叠地址(仅对于311-341 CPU)

当使能输入触点%Q0014为ON时，从存贮器%M0001到%M0033存贮器移动48位，即使目标地址重叠源地址16位，移动仍能够正常完成(除了前述的351和 352 CPU)。

```
|
|%Q0014  _____
|——| |———|MOVE_|—
|        |     |
|        | WORD|
|        |     |
|%M0001 —|IN  Q|—%M0033
|        |     |
|        | LEN |
|        |00003|
|        |_____|
|
```

使用MOVE功能之前:

输入 (%M0001 到 %M0048)

																1
%M0016	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
%M0032	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
%M0048	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

使用MOVE之后:

输出 (%M0033 到 %M0080)

																33
%M0048	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
%M0064	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
%M0080	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

示例 2 – 对于所有CPU

在这个例子中，只要%I0003为ON，在3个位中%M0001, %M0002, 和%M0003的值移动到%M0100, %M0101, 和 %M0102中，另外,线圈 %Q0001变为 on.

```
|
|%I0003  _____                                                    %Q0001
|——| |———|MOVE_|——————————————————————————————————————————————————( )—
|        |     |
|        | BIT |
|        |     |
| %M0001—|IN  Q|—%M0100
|        |     |
|        | LEN |
|        |00003|
|        |_____|
|
```

BLKMOV (INT, WORD, REAL)

块移动(BLKMOV)功能用来复制7个常量作为一个块到指定的位置。

注意

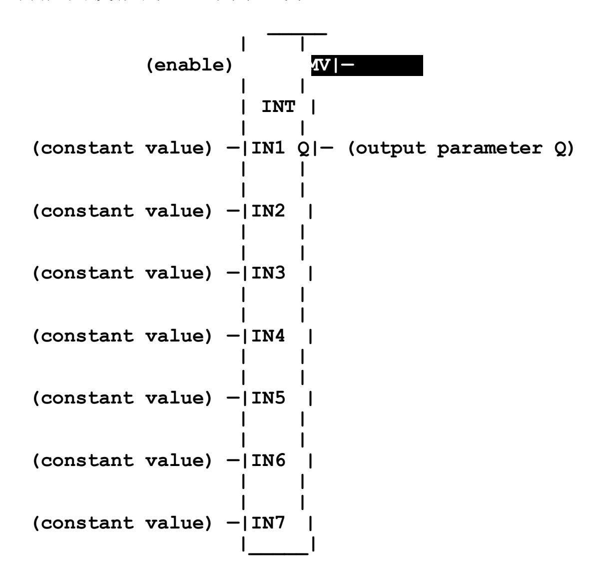
REAL数据类型仅对于35x 和 36x 系列 CPU, 9 或更高版本, 和系列352和37x. CPU所有版本有效。

BLKMOV有8个输入参数和2个输出参数，当功能块就通电流时，它将常量值复制到输出（Q）目标指定起始连续的位置中，输出Q不能作为其他功能的输入。

注意

对于BLKMOV_INT, IN1 — IN7 的值显示位带符号十进制形式，对于BLKMOV_WORD, IN1 — IN7 显示为16进制. 对于 BLKMOV_REAL, IN1— IN7 显示为实数形式

只要功能块使能为ON即向右传递电流。.



参数

参数	说明
使能enable	当功能被使能时，则执行该功能。.
IN1— IN7	IN1 到IN7即7个常量.
ok	当功能块使能时，OK产生输出
Q	输出 Q被移动数组的首个整数。 IN1 被移动到Q.

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN1 — IN7											•	
ok	•											•
Q		•	•	•	•	o†	•	•	•	•		

注意: 对于REAL型数据, 仅对 %R, %AI, 和 %AQ有效.

• 当电流通过功能时的有效参考地址或位置。

o 仅仅对 WORD 型数据有效;对于 INT 或 REAL型无效.

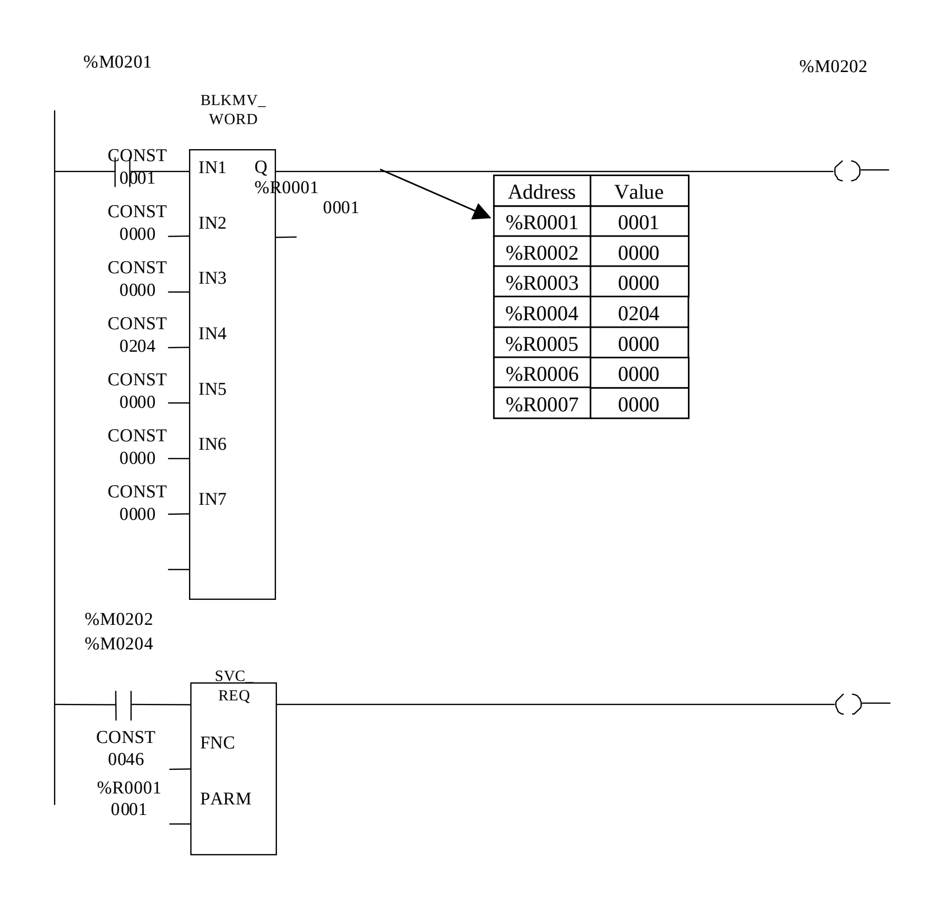
† 仅对%SA, %SB, %SC有效; 不能使用%S

注意

浮点数类型仅使用于35x 和 36x 系列 CPU, 9 或更高版本, 和CPU352与37x的所有版本. 90-30系列CPU仅仅可以使用BLKMV_REAL.

示例

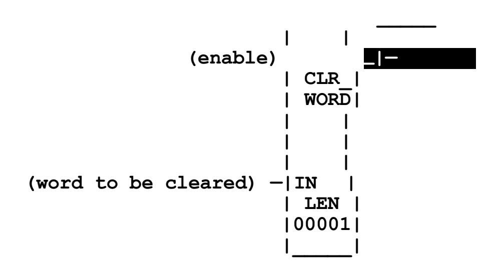
在下例中，当输入使能触点%M0201为ON时，BLKMOV复制7个输入常量到存贮器%R0001 (Q中指定地址)到%R0007中，如果BLKMV成功执行，输出OK %M0202将为ON，换言之，触点%M0202在下一行作为使用%R0001 到 %R0007作为它的参数的SVC_REQ功能的使能(更多信息参看第十二章服务请求指令)



BLKCLR (WORD)

块清除(BLKCLR)功能用来将指定的块用0来填充。

BLKCLR功能有2个输入参数和1个输出参数，当功能块接通电流时，它将0写入到输入IN为指定起始地址的存贮器中，当数据从(%I, %Q, %M, %G,或 %T)存贮器中被清除时，于与参考地址关联的转换信息也被清除，功能块向右传送电流。



参数

参数	说明
使能enable	当功能块被使能时，清除数组。
IN	IN即为要清除的数组的起始地址ed.
ok	当功能块使能时，OK产生输出。.
LEN	LEN 在1－256之间

有效存贮器类型

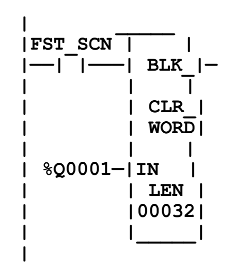
参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN		•	•	•	•	•†	•	•	•	•		
ok	•											•

• 当电流流经功能模块时的有效参考地址和位置.

† 仅适用%SA, %SB, %SC;不能使用 %S 。

示例

在下例中，当电源通电，%Q存贮器%Q0001起始的32个字（512点）被0填充。



SHFR (BIT, WORD)

使用移位寄存器(SHFR)功能模块可以移动一个或多个数据字或数据位从一个指定位置到一个指定存贮区域内。例如，一个字可以移位到一个指定5个字长的存贮区域，作为这次移位的结果，另一个字将从存贮区的末端移出。

注意

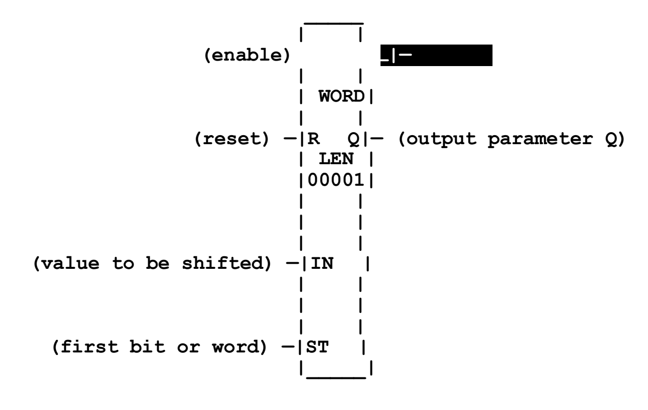
当分配参考地址时，在多字功能模块中的输入和输出出现重叠时，会产生不可预想的后果。

移位功能模块有4个输入参数和2个输出参数，复位输入（R）优先于功能模块的使能输入，当复位激活时，从LEN指定的长度的移位寄存器（ST）开始的所有指定的地址将填充0。

如果功能接通电流并且复位没有激活,移位寄存器中的每一位或字将移向下一个最高位参考地址，移位寄存器中的随后单元移入到Q，如果Q有特别的地址，那么数据将移出Q舍弃，然而，如果IN和Q给定相同的地址，数据将在移位寄存器中循环，IN移位寄存器单元的最高给定地址将移入从ST起始的所空出的单元内，移位寄存器中的内容可以全部进入程序，因为他们被可寻址逻辑存贮器的绝对地址所覆盖。

只要功能块通过使能逻辑得电，即向有传送电流。

当功能模块使能为ON时在每个扫描周期将执行一次,所以它可以使用从跳变线圈得脉冲信号的使能触点，如果它想给定触点得电一次而进行一次移位。



参数

参数	说明
使能enable	当 使 能 输 入 为 ON并 且 R为 OFF时， 功 能 块 将 执 行， 注 意 SHFR 将在使能时每次扫描执行一次
R	当R 输入为ON时，在ST出的移位寄存器将被0填充。
IN	IN 移位寄存器的的首个位或字，对于SHFR_BIT ，所有离散地址都可以使用，不需要排序，然而，在在线时显示16位指定参考地址的起始地址
ST	ST移位寄存器的的首个位或字，对于SHFR_BIT ，所有离散地址都可以使用，不需要排序，然而，在在线时显示16位指定参考地址的起始地址
ok	只要输入使能为ON，R为OFF时，输出OK变产生输出。.
Q	输出Q即从移位寄存器移出的位或字，对于SHFR_BIT 来说，任何离散量都可以使用，且不需要排序，然而，在在线时显示16位指定参考地址的起始地址
.LEN	LEN 决定移位积存器的长度，对于SHFR_WORD, LEN必须在1和256个字之间，对于SHFR_BIT, LEN必须在1到256位之间

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
R	•											
IN		•	•	•	•	•	•	•	•	•	•	
ST		•	•	•	•	•†	•	•	•	•		
ok	•											•
Q		•	•	•	•	•†	•	•	•	•		

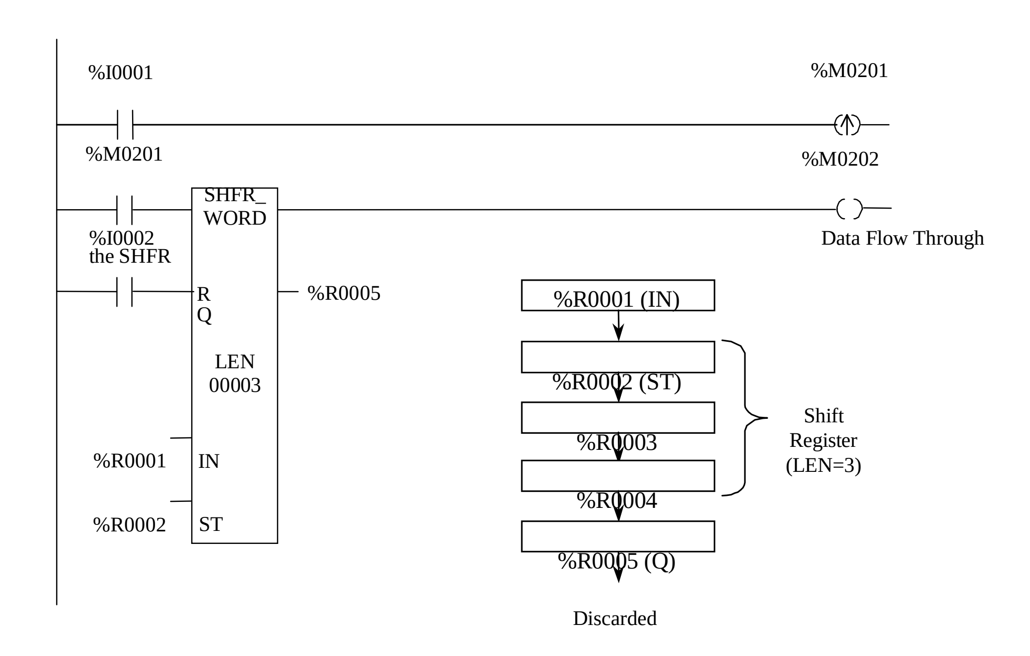
• 当功能模块接通电流时，BIT或WORD有效参考地址或位置。对于SHFR_BIT,离散参考地址可以使用%I, %Q, %M,和 %T，不需要排序。.

† 仅%SA, %SB, %SC 可用; %S 不能用。.

示例 1

在这个例子中，移位寄存器将操作3个（LEN=3）存贮单元%R0002到%R0004，当复位触点%I0002为ON时，3个移位寄存器设置为0。当触点%I0001关闭时，SHFR 功能的使能%M0201闭合一次，在%R0004中数据移位到输出Q的地址%R0005中（%R0005中的数据被舍弃）。地址%R0003中的诗句移位到%R0004，%R0002中的数据移位到%R0003中，和%R0001中的数据移位到%R0002中，数据在下图中显示，如果想得到循环数据，可以通过在输入IN和输出Q使用相同的地址。

.



示例 2

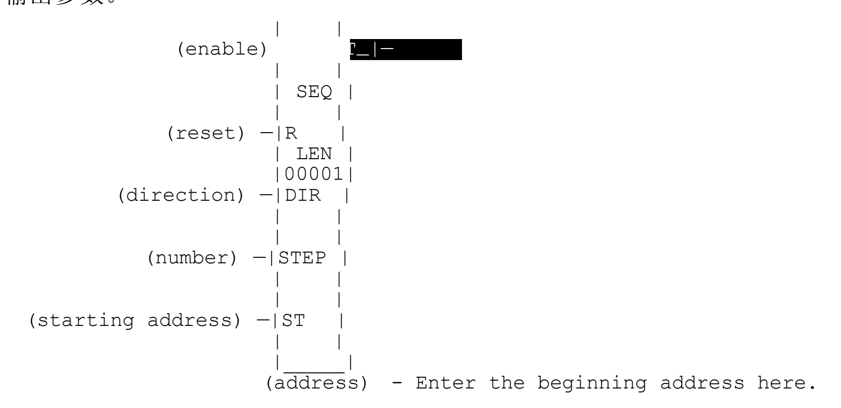
在例2中，移位积存器是BIT类型，长度LEN是100，操作存贮器位置是%M0001到%M0100，当复位参考地址CLEAR激活是，SHFR功能填充0到%M0001到%M0100地址。

当NXT CYC（跳变线圈的脉冲触点）为ON和CLEAR为OFF，SHFR功能将%M0001 到 %M0100中的数据移动一位，当位移出%M0100写入到Q（%M0200）中，在%Q0033中的位被移入到%M0001，Q中的前值被舍弃。

```
|          _____
|NXT_CYC  |     |
|—| |————|SHFR_|—
|         |     |
|         | BIT |
| CLEAR   |     |
|—| |————|R   Q|—%M0200
|         | LEN |
|         |00100|
|         |     |
|         |     |
|         |     |
| %Q0033—|IN    |
|         |     |
|         |     |
| %M0001—|ST    |
|         |_____|
|
```

BITSEQ (BIT)

位定序器(BITSEQ)功能将通过一个位数组执行位定序器功能。BITSEQ功能有5个输入参数和一个输出参数。



使能端要求

位定序器使能输入要求从0到1跳变执行一次，将不再执行直到另一次跳变信号。因此，使用跳变线圈触点作为使能输入端是必需的。

R (复位) 输入

当复位端输入为on, 位定序器功能不执行.

复位端 (R) 优先于使能端 (EN) 且对位定序器功能进行复位. 当R端为ON时,当前步数被设置为步数参数输入的数值且其他所有位设置为0. 如果没有指定STEP (STEP=0), step 设置为1其他所有位为0.

当EN为ON且R为OFF时, 由当前步所指定的位清零. 而当前步数值是增还是减取决于DIR 参数. 然后由新步数指定的位置1.

STEP输入

- 当步数值不断增加并超出范围(1 ϒ⇐ 步数ϒ⇐ LEN), 它被置为1.
- 当步数值不断减小并超出范围 (1 ϒ⇐ 步数ϒ⇐ LEN), 它被置为LEN.

参数ST是可选的. 如果不用(缺省为0), 向上面描述, 没有位被置位或清零，BITSEQ 步数在允许的范围内进行循环.

DIR输入

DIR为ON或OFF可改变移位方向。如果为ON位增加，如果为OFF位减少。

ST和 LEN 参数

ST输入为位序列第一个字。数组长度，按位，由LEN设定。例如，如果ST为%M0001 ，LEN等于16，数组为从 %M0001 到 %M0016. 如果 ST为%R地址, LEN 为数组%R寄存器位数. 例如，如果ST是 %R0004, LEN 等于8，该数组为%R前8位; %R0004被忽略r.

位定序器的内存要求

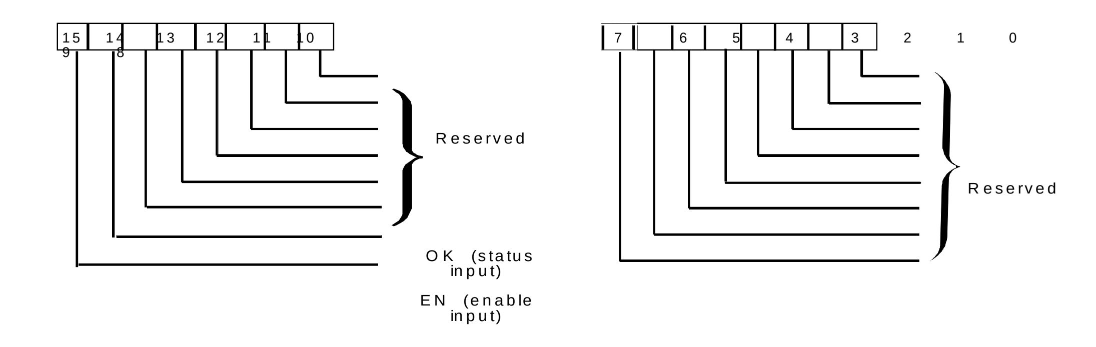
每个位定序器需使用三个%R 寄存器存储以下数据:



当前步数	word 1
位序列长度 (位)	word 2
控制字	word 3

当你使用Logicmaster编程时,对于位定序器功能你必须在此功能块的图块下方输入这三个连续字的地址.

控制字用于存储与此功能块有关的布尔输入输出状态，如下面格式:



注意

0到13位不用. 注意STEP为1到16，不是0到15。

参数

参数	说明
address	当前步数，位序列长度，控制字存储单元.
enable	使能为ON，如果上一次扫描时使能为OFF且R为OFF,则执行位序列功能.
R	当R为ON, 位序列中的步数被置为STEP 中的值(默认为1), 除了当前步，其余位均清零.
DIR	当DIR为ON时，位序列数在移位前增加，否则就先减小.
STEP	当R为ON，步数被设置为该值.
ST	ST 包含位序列第一个字.
ok	一旦选通该功能块，OK为ON.
LEN	LEN必须为1到256.

注意

在BITSEQ功能块进行线圈检查时，将从ST开始检查16位，即使LEN的值小于16.

有效寄存器类型

Parameter	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
address								•				
enable	•											
R	•											
DIR	•											
STEP		•	•	•	•		•	•	•	•	•	•
ST		•	•	•	•	•†	•	•	•	•		•
ok	•											•

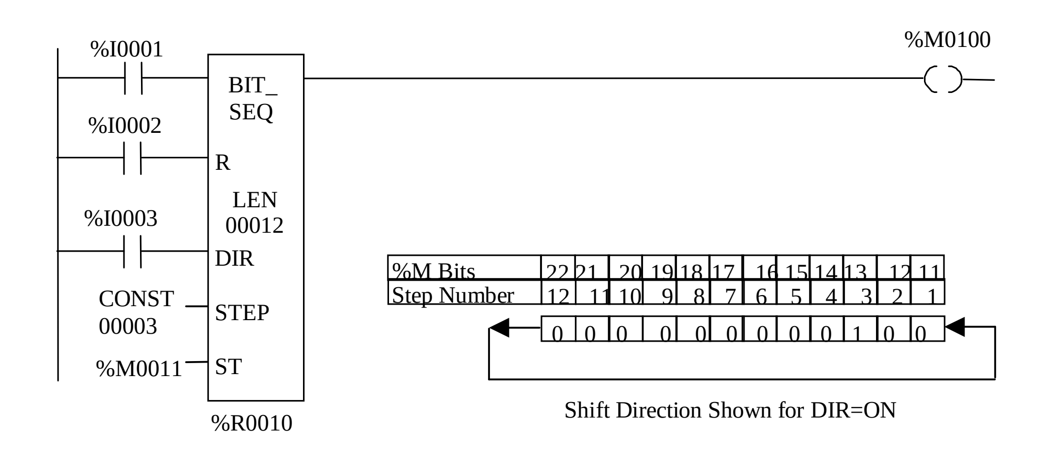
• Valid reference or place where power may flow through the function.

† SA, %SB, %SC only; %S cannot be used

示例

本例中，位定序器对%M0011 (ST)到%M0022 (长度等于12)进行操作. 控制字存储在%R0010, %R0011,和 %R0012中. 当 %I0002 (on the R input) 为on, 定序器复位，表示步数（STEP）3对应的位为1其他所有位为0.

当%I0001为1 (%I0002为off), 步数3对应的位清零，如果DIR为ON步数4对应的位置1，或如果DIR为OFF步数2对应的为置1。



%M Bits	22	21	20	19	18	17	16	15	14	13	12	11
Step Number	12	11	10	9	8	7	6	5	4	3	2	1
	0	0	0	0	0	0	0	0	0	1	0	0

COMMREQ

如果程序需要与智能模块像Genius通讯模块或可编程协处理器通讯时，可以使用通讯功能模块(COMMREQ).

注意

在下页中的介绍是显示COMMREQ 功能的常规格式，对于每种类型的设备需要增加 COMMREQ 功能的编程信息在模块的文档里。

COMMREQ功能有3个输入参数和1个输出参数，当COMMREQ接通电流是，数据通讯块被送到智能模块，指令模块从参数IN所标定的给定地址开始，在SYSID中标定智能模块的机架号和槽号。

COMMREQ功能可以发送信息并等待回复或发送信息不等待回复而继续执行,如果指令模块指定程序不需要回复，指令块的内容送到装置而程序立即（暂停时间可忽略不计）执行，这就是**NOWAIT模式。**

如果指令模块指定程序要等待一个回复，则指令模块的内容送到设备后CPU等待一个回复，PLC等待设备作出回应的最长时间规定在指令块中，在此时间内，如果设备未作出响应，程序将继续执行，这就是**WAIT模式。**

功能块的故障（FT）入轨在下列情况下可能是ON：

1. 指定目标不在 (SYSID) 指定位置.
2. 指定任务 (TASK) 数字对于设备无效
3. 数据长度是0（在指令块中）
4. 设备状态点的地址不存在（指令块中），这可能导致存贮器选型不正确，或存贮器内的地址超出范围。

指令模块

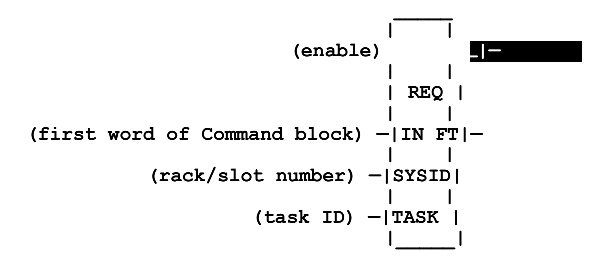
指令模块提供目标智能模块的信息，包含执行交换数据的指令数目。

指令模块的地址规定用作在COMMREQ功能模块中的输入IN，该地址可以处于存贮器（%R, %AI, 或%AQ）区域的任何字，指令块的长度取决于传送设备的数据量。

指令块具有如下结构：

长度(以字为单位)	address
等待/不等待 标志	address + 1
状态指针寄存器	address + 2
状态指针偏移量	address + 3
闲置超时值	address + 4
最长通讯时间	address + 5
数据块	address + 6 to address + 133

指令模块所需要的信息可利用适当的编程功能如块移动或移动系列功能放在所设定的存贮区内。



参数

参数	说明
使能enable	当使能输入为ON时，通讯模块在每次扫描时执行，如果不是COMMREQ功能多次传送时，使能输入应该是转换线圈的触点。.
IN	IN 表示命令块第一个字.
SYSID	SYSID 表示目标设备的机架号 (最高有效位字节) 和槽号(最低有效位字节) .
TASK	TASK 表示任务ID .
FT	如果执行COMMREQ功能出现错误时，FT输出.

注意

90-30系列COMMREQ 没有OK 输出

有效存贮器类型

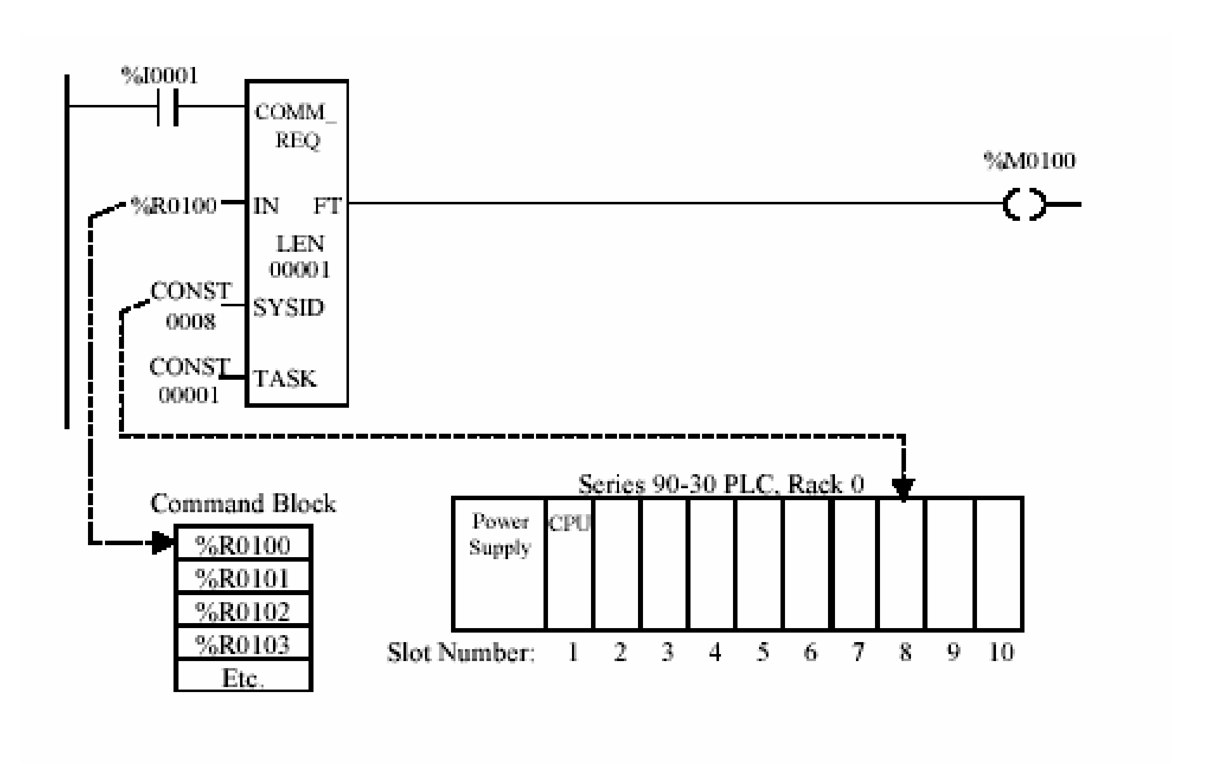
参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN								•	•	•		
SYSID		•	•	•	•		•	•	•	•	•	
TASK								•	•	•	•	
FT	•											•

• Valid reference or place where power may flow through the function.

示例

本例中，使能%I0001为on，命令块起始地址%R0100(IN输入指定)发送给0号机架8槽(SYSID=0008)模块，通讯任务为1(TASK = 1)。如果出现错误，FT输出ON，使得%M0100为ON。

注意IN指定命令块起始地址。 SYSID十六进制，指定目标模块机架号和槽号; 高字节表示机架号低字节表示槽号。 因此，示例中SYSID 为0008表示机架00和槽08. 机架0永远代表主或CPU机架，因此如果目标模块在扩展机架或远程机架，高字节将不是0而是目标模块所在得机架号。



Chapter 10 第十章 数据表格功能模块

数据表格指令用来执行下面的功能：

缩写	功能	说明	页码
ARRAY_MOVE	数组传送	从源数组到目标数组复制一定数量的数据单元。	10-2
SRCH_EQ	查寻相等	查寻等于指定数值的所有数组值。.	10-7
SRCH_NE	查寻不相等	查寻不等于指定数值的所有数组值。.	10-7
SRCH_GT	查寻大于	查寻大于指定数值的所有数组值。	10-7
SRCH_GE	查寻大等于	查寻大于或等于指定数值的所有数组值。	10-7
SRCH_LT	查寻小于	查寻小于指定数值的所有数组值。.	10-7
SRCH_LE	查寻小等于	查寻小于或等于指定数值的所有数组值	10-7

对于这些功能模块，最大长度是32,767 个字节或 字 或 262,136位 (位只使用于 ARRAY_MOVE).

数据表格功能模块按下述数据类型操作：

数据类型	说明
INT	带符号整数
DINT	双精度带符号整数
BIT *	位数据类型
BYTE	字节数据类型
WORD	字数据类型

* 仅使用于 ARRAY_MOVE.

缺省的数据类型是带符号整数，在梯形逻辑软件中选定制便功能后数据类型可以改变，对比两种数据类型或两种不同的数据类型，首先选用适当的转换功能（在第十一章“转换功能模块”中所述）将数据转换成上表中所列的数据类型之一。

ARRAY_MOVE (INT, DINT, BIT, BYTE, WORD)

数组和数据单元的定义

这部分讨论的是，数组是PLC存贮器中一组连续的地址如%R0100 到 %R0120，数据单元是占据在数组内存中的一个单元，例如，如果一个数组是位类型的，那么每个单元占据存贮器中一个单独的位，像%M0001（或可能是指定存贮器中一个独立的位），或者数组是字类型，那么每个数据单元就是存贮器中16位的字，如%R0100（也可能是16个连续的%I位），关于这个更多的信息参看“有效存贮器类型”表。

指针代码

每个数据单元在数组中都有指定的号码称为指针代码，被PLC自动识别。指针代码表示的是数据单元在数组中的位置，数据单元在数组中向上编码，起始于存贮器最低位，标记指针代码为1。

例如，在下面起始地址为%R0105的字类型的数组中，它有指针代码为1到10的10个数据单元。

Address	Index No.
%R0105	1
%R0106	2
%R0107	3
%R0108	4
%R0109	5
%R0110	6
%R0111	7
%R0112	8
%R0113	9
%R0114	10

Array Move 指令

数组移动功能用来从源数组到目标数组复制一定数量的数据单元。每个由Array Move功能指定的参考地址都有相等数量的数据单元，Array Move功能允许源数组与目标数组不同的相关位置之间进行，例如，在源数组中起始指针为5的3个数据单元可以复制到目标数组的起始指针为7的3个数据单元中。

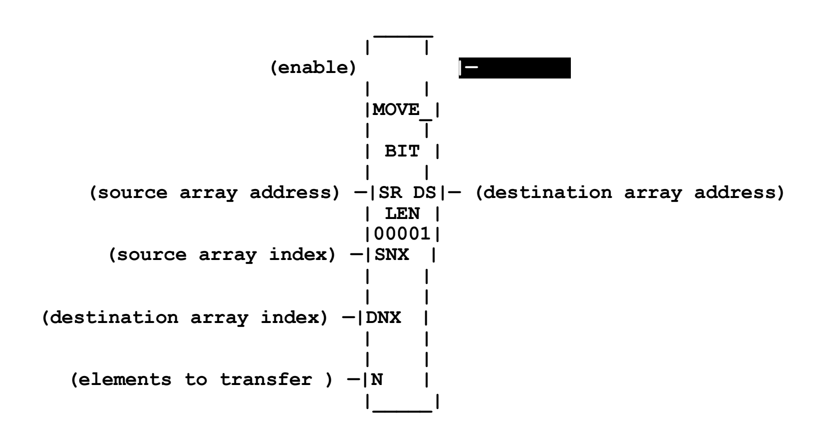
ARRAY_MOVE功能有5个输入参数和2个输出参数，当功能被使能时，在计数指示器（N）中的数据单元的数目将从输入数组中从指针起始位置为SNX的输入复制，这些数据写入到起始指针位置为DNX的输出数组中，LEN指的是每个数组中指定的操作数的数量。

对于ARRAY_MOVE_BIT, 当字定向存贮器选作源数组的参数和（或）目标数组的起始地址时，不管数组长度如何，在编程器窗口中以16位显示。

ARRAY_MOVE 指令中的索引都是基于1的. 在使用ARRAY_MOVE时, 有起始地址和长度确定的源数列和目的数列以外的元素不能被赋予变量地址。

OK输出将接通，除非出现下述情况：

- 使能置为 OFF.
- (N + SNX – 1) 大于 LEN. 这个公式用来由PLC确认没有超出指定的源数组。
- (N + DNX – 1) 大于 LEN. 这个公式用来由PLC确认没有超出指定的目标数组。
- SNX 或 DXN = 0.



参数

参数r	说明
enable	当使能接通时，数组移动功能执行.
SR .	SR即源数组的起始地址，对ARR-MOVE-BIT来说，任何参考地址均可用，无需排序字节，然而从所指定的参考地址的16位则在编程器中显示。
SNX	SNX 即源数组的复制的首单元的指针代码。.
DNX	DNX即目标数组的复制的首单元的指针代码。.
N	复制的数据单元的数量。
ok	当使能接通时，OK则有输出.
DS	DS 即目标数组的起始地址，对于 ARRAY_MOVE_ BIT，任何参考地址均可用，无需排序字节，然而从所指定的参考地址的16位则在编程器中显示
LEN	LEN起始与SR和DS,形成每个数组的单元数量

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
SR		o	o	o	o	–†	o	•	•	•		
SNX		•	•	•	•		•	•	•	•	•	
DNX		•	•	•	•		•	•	•	•	•	
N		•	•	•	•		•	•	•	•	•	
ok	•											•
DS		o	o	o	o	†	o	•	•	•		

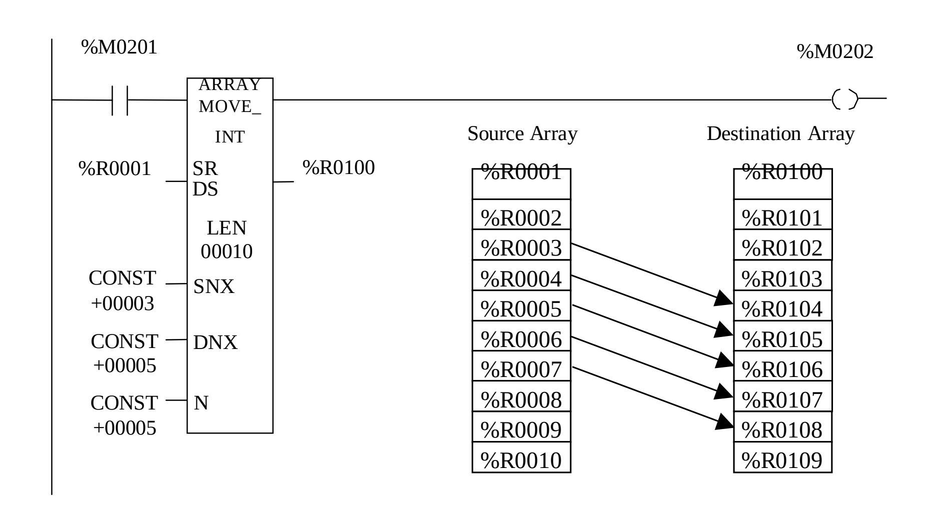
• 功能模块通流时的有效地址和位置，对ARRAY_MOVE_BIT, 离散使用的参考地址 %I, %Q, %M, 和%T不需排序字节。仅对INT, BIT, BYTE, 或 WORD 数据有效 ，不适用 DINT数据类型。.

– 仅对INT, BIT, BYTE, 或 WORD 数据有效 ，不适用 DINT数据类型。

† 仅对 %SA, %SB, %SC;不能使用%S

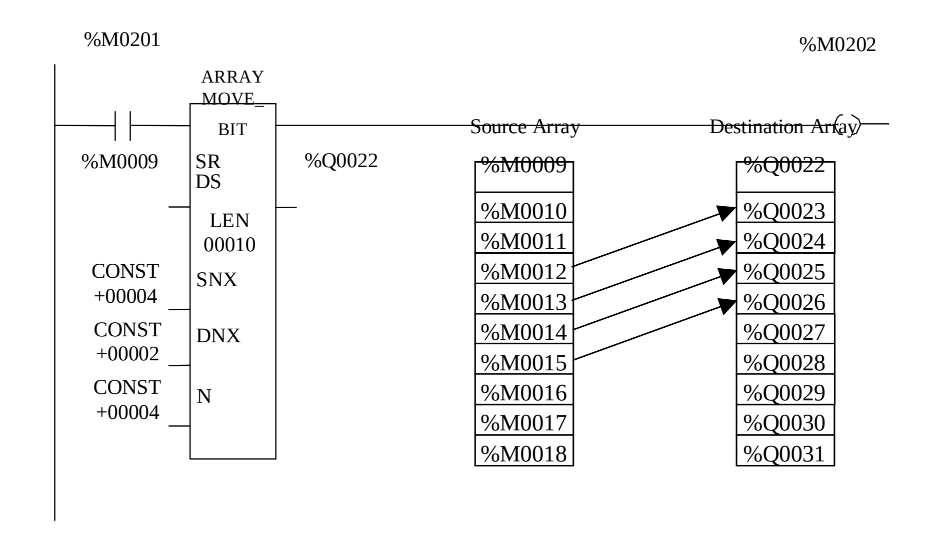
示例 1

在这个例子中，两个数组都是INT类型并且都有10个单元（整数），通过LEN=10指定，在SR和DS中指定他们的起始地址，当使能触点%M0201为ON时，从源数组到目标数组拷贝5个（N=5指定）数据单元，源数组5个被复制的数据单元的起始指针代码为3，因为SNX=3，拷贝到目标数组起始指针代码为5（DNX=5）的5个数据单元，所用读取源数组的%R0003到%R0007拷贝到目标数组的%R0104到%R0108中。



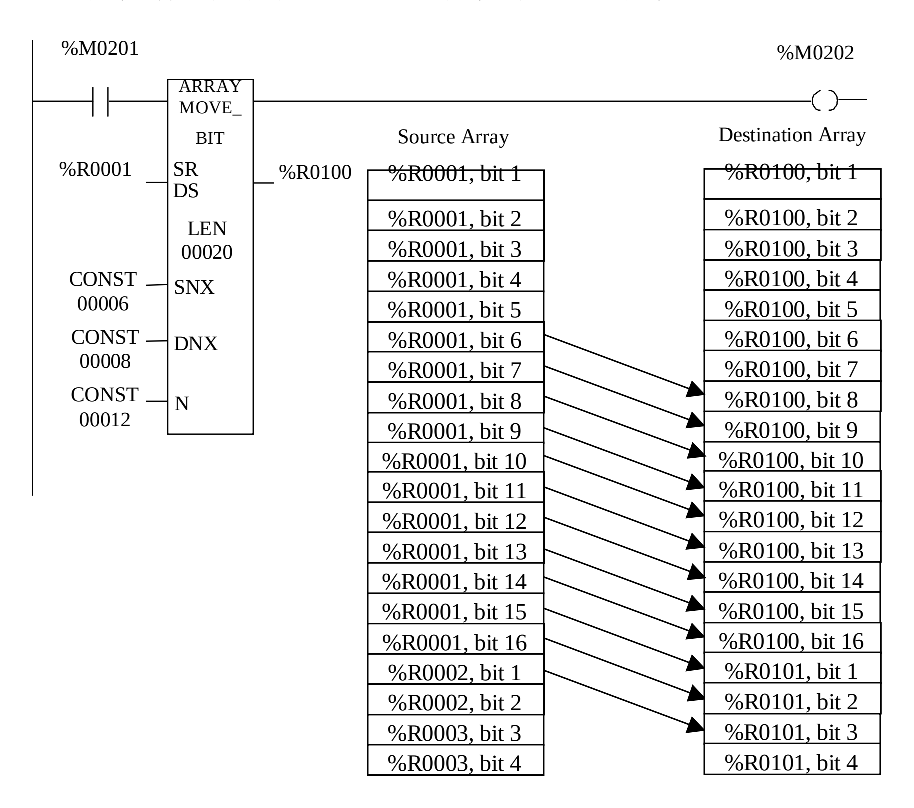
示例 2

在这个例子中，两个数组都是INT类型并且都有10个单元（整数），通过LEN=10指定，在SR和DS中指定他们的起始地址，当使能触点%M0201为ON时，从源数组到目标数组拷贝4个（N=4指定）数据单元，源数组4个被复制的数据单元的起始指针代码为4，因为SNX=4，拷贝到目标数组起始指针代码为2（DNX=2）的4个数据单元，所用读取源数组的%M0012到%M0015拷贝到目标数组的%Q0023到%Q0026中。



示例 3

在这个例子中，两个数组都是INT类型并且都有20个单元（整数），通过LEN=20指定，在SR和DS中指定他们的起始地址，当使能触点%M0201为ON时，从源数组到目标数组拷贝12个（N=12指定）数据单元，源数组12个被复制的数据单元的起始指针代码为6，因为SNX=6，拷贝到目标数组起始指针代码为8（DNX=8）的4个数据单元，所用读取源数组的%R0001中6位到%R0002中1位拷贝到目标数组的%R0100中8位到%R0101中3位。



查寻功能

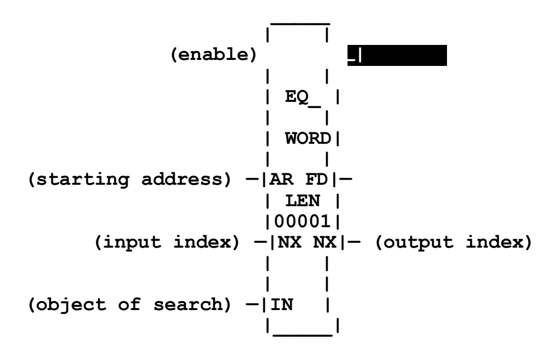
使用下表中所列查寻功能，可以为特殊操作查寻所有数组的数值。

缩写	功能	说明
SRCH_EQ	查寻相等	查寻数组里等于指定值的值
SRCH_NE	查寻不等l	查寻数组里不等于指定值的值.
SRCH_GT	查寻大于	查寻数组里大于指定值的值.
SRCH_GE	查寻大于或等于	查寻数组里大于或等于指定值的值
SRCH_LT	查寻小于	查寻数组里小于指定值的值.
SRCH_LE	查寻小于或等于l	查寻数组里小于或等于指定值的值

查寻功能有4个输入参数和2个输出参数，当功能块接通电流时，数组从起始(AR + 输入NX)位置查寻，这个起始地址是数组（AR）加上数组的指针（输入NX）。

查寻直到找到查找目标（IN）中的数组单元或直到查寻完毕，如果发现数组单元，输出参数（FD）将被置为ON，如果在查寻完毕之前没有发现数组单元，输出参数（FD）置为OFF并且输出参数（输出NX）被置为0。

输入NX的数值在0－LEN-1之间有效，NX在开始查寻第一个单元时被设为0，在执行时间这个值的增加1，然而，输出NX的值是1到LEN，如果输入NX值超出风味（<0或>LEN）它的值将被默认为0。



参数

参数	Description
enable	当使能为ON时，执行此功能。.
AR	AR 查寻数组的起始地址 (目标数组).
输入 NX	输入 NX 开始查寻的指针 (在目标数组中)
IN	IN 要查找的目标.
输出 NX	如果查找到目标，则写入数组中的位置 (它的指针代码)
FD	FD的输出表示在数组中找到目标单元。
LEN	LEN 将指定被查寻单元从AR起始的单元数，它可以是1到32767个字节或字。

有效存贮器类型

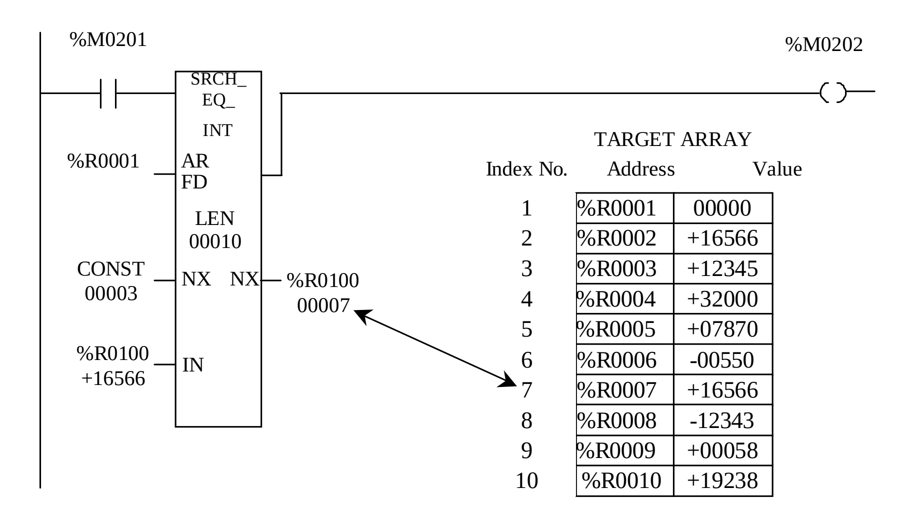
参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
AR		o	o	o	o	–	o	•	•	•		
NX in		•	•	•	•		•	•	•	•	•	
IN		o	o	o	o	–	o	•	•	•	•	
NX out		•	•	•	•			•	•	•	•	
FD	•											•

• 功能模块通流时的有效参考地址或位置.
o 只用于 INT, BYTE, 或WORD 数据类型;对于 DINT型无效.
– 只对 BYTE或 WORD有效;不适用INT或DINT类型.

示例 1

查寻功能模块（INT类型）在这个例子中查寻存贮器块中起始地址%R0001开始，到%R0010（LEN=10）在IN中定义＋16566作为被查寻的值，输入NX的值是3，表示在数组中从第四位数据单元开始查寻，直到当动能块执行完成NX的值增加为1。

当使能触点%M0201为ON时，SRCH_EQ功能在指定的数组中查找，起始于指针代码为4，等于在输入IN中＋16566的单元，在%R0007，指针代码为7，中发现这个数值，所以写入7到输出NX中在%R0100地址，它也将输出FD置位，说明在这个数组中找到目标，注意尽管在%R0002地址中也包含需要找的数值+16566，因为在输入NX参数的数值是3，指定查找从第四位%R0004开始,所以这个数据单元不包括在查找单元中。



Index No.	Address	Value
1	%R0001	00000
2	%R0002	+16566
3	%R0003	+12345
4	%R0004	+32000
5	%R0005	+07870
6	%R0006	-00550
7	%R0007	+16566
8	%R0008	-12343
9	%R0009	+00058
10	%R0010	+19238

示例2

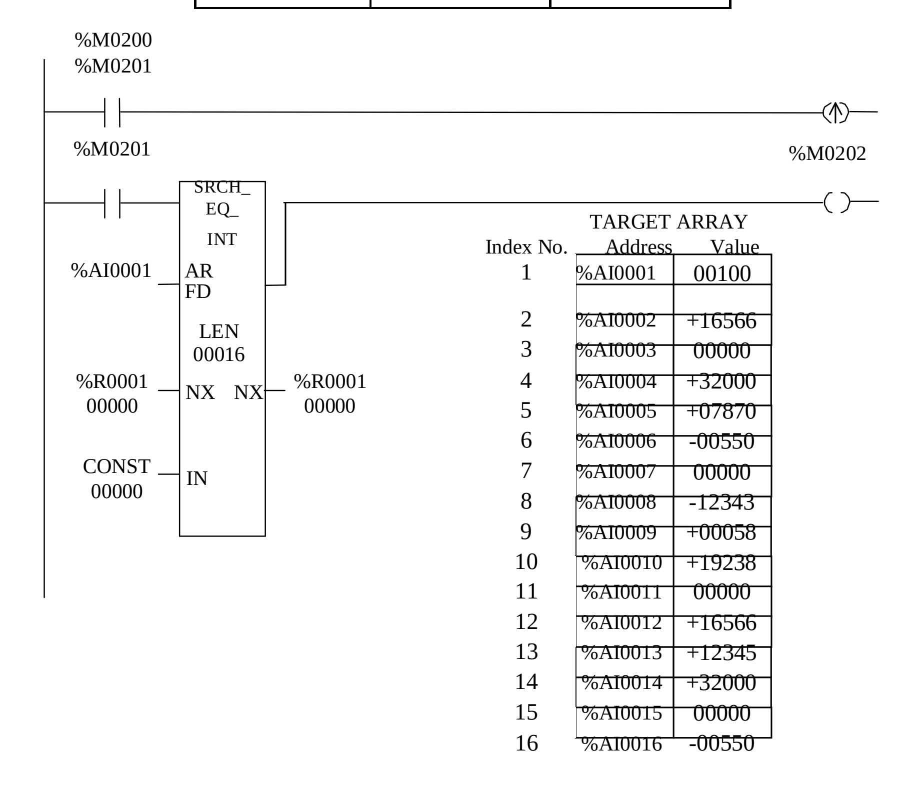
在这个例子中，数组的起始地址为%AI0001（在AR中指定）到%AI0016（LEN=16）在IN中定义＋16566作为被查寻的值，输入NX的值是0，表示在数组中从第一位数据单元开始查寻，直到当动能块执行完成NX的值增加为1。

当%M0200第一时间关闭时，功能模块执行第一次查寻，起始于数据单元为1，SRCH_EQ功能在指定的数组中查找，起始于指针代码为4，等于在输入IN中＋16566的单元，在%R0007，指针代码为7，中发现这个数值，所以写入7到输出NX中在%R0100地址，它也将输出FD置位，说明在这个数组中找到目标，注意尽管在%R0002地址中也包含需要找的数值+16566，因为在输入NX参数的数值是3，指定查找从第四位%R0004开始,所以这个数据单元不包括在查找单元中。

当%M0200第一时间关闭时，功能模块执行第一次查寻，起始于数据单元为1，查寻等于在IN，00000，处的值，在%AI0003指针代码为3找到相等的值在%AI0003指针代码为3找到相等的值，所以将在同样拥有%R0001参考地址的输出NX和输入NX写入3,输出FD变为ON，表示在数组中找到目标值。

当%M0200第二此关闭时，输入NX 的值现在被设定为3，增量为1，所以查找从第四个数据单元%AI0004开始目标值在%AI0007中被找到，第7个数据单元，所以数字7被写入到%R0001中.。延续这种路径进行每次成功的查找，直到第15次查找，没有找到目标值时，在%R0001中写入0。

Search No.	Search Starts at Data Element	Search Results (in %R0001)
1	1	3
2	4	7
3	8	11
4	12	15
5	16	0



Chapter 11

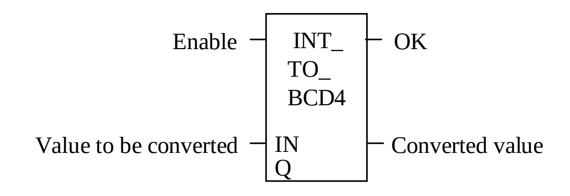
第十一章 转换功能模块

转换功能模块用来将数据单元从一种数据类型转换成另一种数据类型，一些程序指令像数学功能模块必须使用同类型的数据，这部分将介绍如下转换功能模块：

—>BCD-4 (INT)

BCD-4功能用来将带符号整数转换成等值的4位BCD码输出.功能块不能改变原来的数据，数据可以转换程BCD格式用以驱动BCD编码的LED显示或预置外部设备，如高速计数器。

当功能块接通电流时，将执行转换，通过输出Q输出有效的结果数值，当功能块接通电流时功能块即通流，除非指定转换的数值在0~9999范围以外。



参数

参数	说明
使能enable	当功能块被使能时，执行转换功能。.
IN	IN 转换成BCD码的整数值的指定地址。.
ok	当功能块无错误执行后，OK便有输出。
Q	Q为IN处转换成BCD的码的输出

有效存贮器类型

参数r	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN		•	•	•	•		•	•	•	•	•	
ok	•											•
Q		•	•	•	•		•	•	•	•		

• 功能模块通流时的有效参考地址或地点

示例

在下例中，当输入%I0002被置位时并且无错误，在本地输入%M0017通过地址%M0032转换成4位BCD码并且结果存贮在%Q0033到%Q0048内，线圈%M0032变成ON，以验证成功进行转换。

```
|           ______
|%I0002    |      |                                                  %M0032
|—| |——————| INT_ |——————————————————————————————————————————————————( )—
|          | TO_  |
|          |      |
|          | BCD4 |
| %M0017———|IN   Q|—%Q0033
|          |______|
|
```

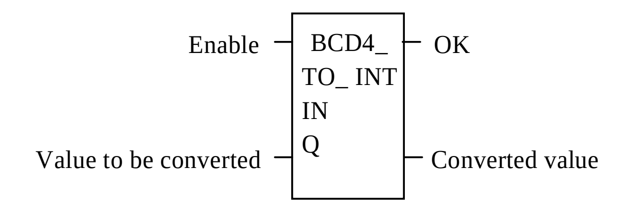
—>INT (BCD-4, REAL)

转换带符号整数功能模块用来输出于BCD-4或REAL型数据等值的整数。这个功能不能改变原来数据。

注意

REAL数据类型那个仅仅使用于35x 和36x 系列 CPU，9 或更高的版本，和 CPU352 、 CPU37x型CPU所有的版本。.

当功能块接通电流时，即执行转换，将有效结果输出到Q中，这个模块在接通电流时通常通流，除非数据超出范围。



参数

参数	说明
使能enable	当使能为ON时，功能被执行。
IN	IN 即转换成整数的BCD-4, REAL,或常量的参考地址。
ok	只要使能产生，OK输出就接通，除非数据超出范围或为NAN（非数值）。
Q	Q即IN处输入原数值转换的整数格式。

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN		•	•	•	•		•	•	•	•	•	
ok	•											•
Q		•	•	•	•		•	•	•	•		

注意: 对于 REAL型数据，仅适用%R, %AI,和 %AQ数据类型.

• 功能模块通流时的有效参考地址或位置.

示例 1 – BCD4转换成INT

在下例中，只要输入%I0002被置位，在PARTS处的BCD-4数值就被转换成带符号整数并被放置在%R0001中，后面的ADD功能模块，%R0001加到指定参考地址RUNNING的带符号整数中，加法和通过ADD模块输出到指定地址TOTAL中。

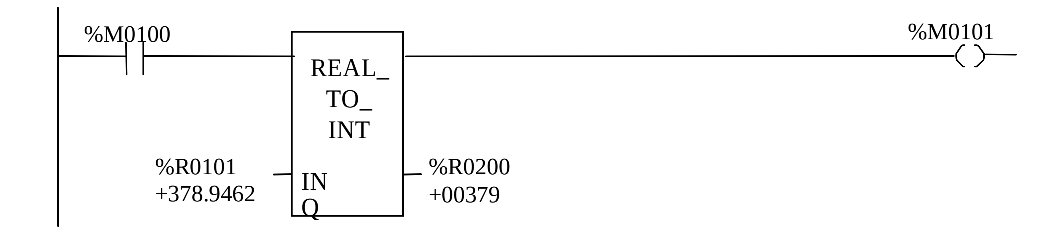
.

```
|
|%I0002  _____                   _____
|--| |---|BCD4_|-----------------| ADD_|-
|        |     |                 |     |
|        | TO_ |                 | INT |
|        |     |                 |     |
|        | INT |                 |     |
| PARTS -|IN  Q|- %R0001  %R0001 |I1  Q|- TOTAL
|        |_____|                 |     |
|                       RUNNING-|I2   |
|                                |     |
                                 _____
```

示例 2 – Real 转换成 Integer

这个例子显示的是在%R0101中的实数转换在%R0200中的整数，当使能输入%M0100为ON是，转换替换，注意在转换过程中，实数被四舍五入到最接近的整数，如果十分位小数是0.5或更大，则结果整数将四舍五入为1，如果十分位小数小于0.5，则十分位舍弃并整数不四舍五入，在下例中，实数378.9462被四舍五入位整379。

如果不想四舍五入，使用REAL_TRUN_INT功能模块，他将直接转换成实数的整数部分，在转换中不理会它的数值。



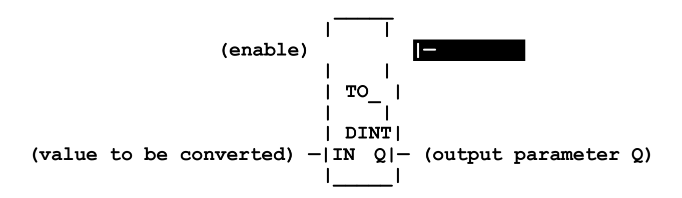
—>DINT (REAL)

转换成双精度整数功能用来输出等值实数的双精度整数，原数据不能被功能模块改变。

注意

REAL数据类型那个仅仅使用于35x 和36x 系列 CPU，9 或更高的版本，和 CPU352 、 CPU37x型CPU所有的版本。.

当功能块接通电流时，即执行转换，将有效结果输出到Q中，这个模块在接通电流时通常通流，除非数据超出范围。



参数

参数	说明
enable	当使能为ON时，功能被执行.
IN	IN 即指定转换成双精度整数的数值的参考地址。
ok	只要使能产生，OK输出就接通，除非数据超出范围
Q	Q即总IN处输入原数值转换的双精度整数格式。

注意

由于REAL有24个有效位，因此当REAL转换成DINT是精度顺失出项是有可能的。

有效存贮器类型

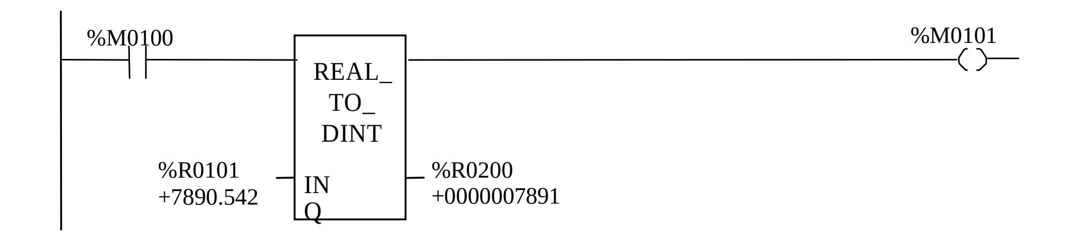
参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN		o	o	o	o		o	•	•	•	•	
ok	•											•
Q								•	•	•		

• 表示定流流过功能块的有效参考地址或位置。.

示例

在下例中，只要使能输入%M0100为ON是，在输入位置%R0101里的实数值转换成双精度整数，并且结果保存在位置%R0200中。注意在转换中，实数被四舍五入到最接近的整数，如果十分位小数是0.5或更大，则结果整数将四舍五入为1，如果十分位小数小于0.5，则十分位舍弃并整数不四舍五入，在下例中，实数7890.542被四舍五入位整7891。

如果不想四舍五入，使用REAL_TRUN_INT功能模块，他将直接转换成实数的整数部分，在转换中不理会它的数值。



—>REAL (INT,　DINT, BCD-4, WORD)

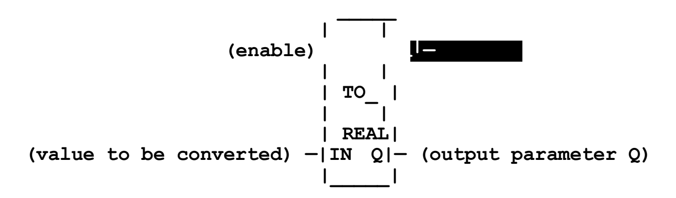
转换实数功能模块用来将输入数据转换成实数输出。这个功能不能改变原来数据。

当功能块接通电流时，即执行转换，将有效结果输出到Q中，这个模块在接通电流时通常通流，除非数据超出范围。

由于REAL有24个有效位，因此当REAL转换成DINT是精度顺失出项是有可能的。.

注意

REAL数据类型那个仅仅使用于35x 和36x 系列 CPU，9 或更高的版本,
和 CPU352 、 CPU37x型CPU所有的版本。



参数

参数	说明
enable	当使能为ON时，功能被执行
IN	IN 即指定转换成实数的数值的参考地址。
ok	只要使能产生，OK输出就接通，除非数据出错
Q	Q即总IN处输入原数值转换的实数格式

有效存贮器类型

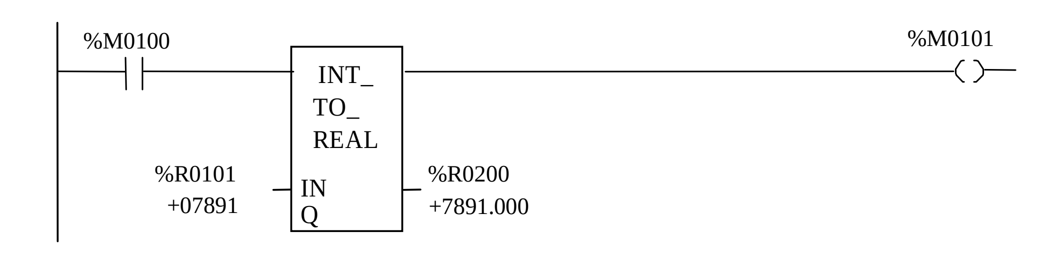
参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN		o	o	o	o		o	•	•	•	•	
ok	•											•
Q								•	•	•		

•　　电流流过功能模块的有效参考地址或位置。

o　　对 DINT_TO_REAL无效.

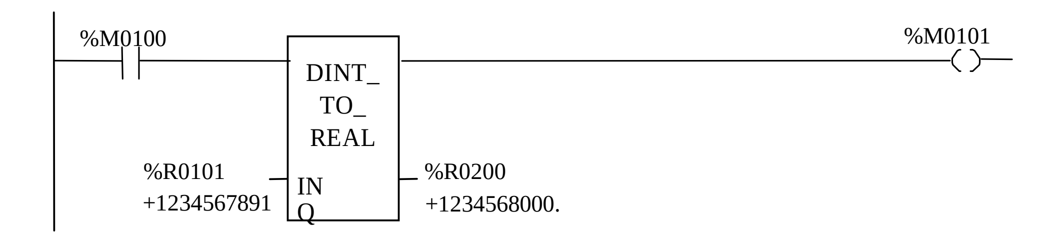
示例 1 –整数到实数的转换

在下例中，IN处输入的整数数值实+07891. 在转换成实数格式为+7891.000后结果放入位置%R0200中。



示例 2 – 双精度整数到实数的转换

在下例中，在IN处输入的双精度整数是 +1234567891，在转换成实数格式为+1234568000后结果放入位置%R0200中。注意一个双精度整数有10有效的位，但是实数却仅仅有7个有效位，因此，在转换实数过程中，整数舍位为7位有效位，例子中显示，双精度整数的最后4位7891被四舍五入位8000，作为实数的最后4位。



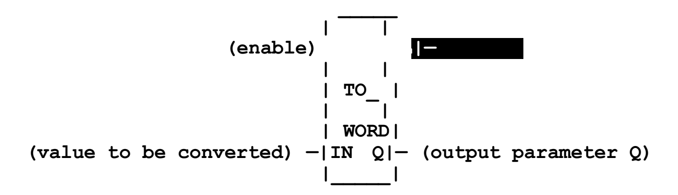
—>WORD (REAL)

转换成字功能用来输出等值实数的字，原数据不能被功能模块改变。

注意

此功能仅使用35x, 36x, 和 37x系列 CPU。

当功能块接通电流时，即执行转换，将有效结果输出到Q中，这个模块在接通电流时通常通流，除非数据超出范围 0 到 FFFFh.



参数

参数	说明
enable	当使能为ON时，功能被执行
IN	IN即指定转换成WORD的数值的参考地址
ok	只要使能产生，OK输出就接通，除非数据出错
Q	Q 即总IN处输入原数值无符号整数格式。

有效存贮器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN								•	•	•	•	
ok	•											•
Q		•	•	•	•		•	•	•	•		

• 表示定流流过功能块的有效参考地址或位置。

示例 – Real到 Word的转换

在这个例子中，因为RANGE功能模块无效于REAL类型，在%R0001中的实数第一次进行转换成一个字（在%R0003中），用来作为接下来RANGE WORD功能的输入。

下表表示的是在下图中输入输出的数值。

Item	Value or State
%R0001	15767.83
%R0003	3A89h (15,768 decimal)
HI LIM	4E20h (20,000 decimal)
LOW LIM	2710h (10,000 decimal)
Q1	ON

```
|
|%I0002  _______                        _______
|       |       |                       |       |
|——|  |———|REAL_ |————————————————————————|RANGE |—
|        |       |                       |       |
|        | TO_   |                       | WORD  |
|        |       |                       |       |
|        | WORD  |                       |       |   %Q0001
| %R0001—|IN   Q |-%R0003     HI_LIM-|L1    Q|———( )————
|        |_______|                       |       |
|                                        |       |
|                              LOW_LIM-|L2      |
|                                        |       |
|                                        |       |
|                               %R0003-|IN      |
|                                       _______
```

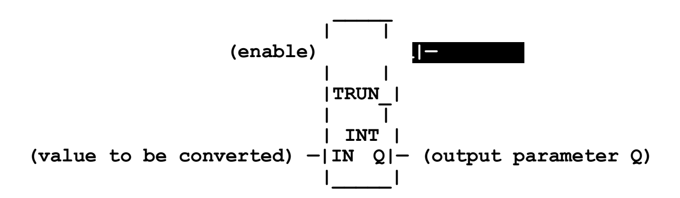
TRUN (INT, DINT)

舍位功能被用来把实数四舍五入为零，在转换过程中，所有数据的右边十分位都被舍位作为输出数，原始数据不能被改变。

注意

35x 和 36x系列 CPU (9.00 或更高版本和 CPU352所有的版本), 和 37x 系列90-30 CPU处理浮点数;因此, TRUN 功能不使用其他90-30 CPU.

当功能块接通电流时，即执行转换，将有效结果输出到Q中，对于CPU 352, 这个模块在接通电流时通常通流，除非数据超出范围或输入IN是NaN (非数值). 对于另外 35x和 36x/37x 系列 CPU，这个功能不能流通电流。.



参数

参数	说明
Enable	当功能块被使能时，功能被执行
IN	IN 即用于舍位的实数参考地址。.
Ok	只要功能块无错执行，OK输出就接通，除非数据超出范围或输入IN是NaN (非数值)
Q	Q 即总IN处输入原数值的舍位格式

注意

由于REAL有24个有效位，因此当REAL转换成DINT是精度顺失出项是有可能的。

有效存贮器类型

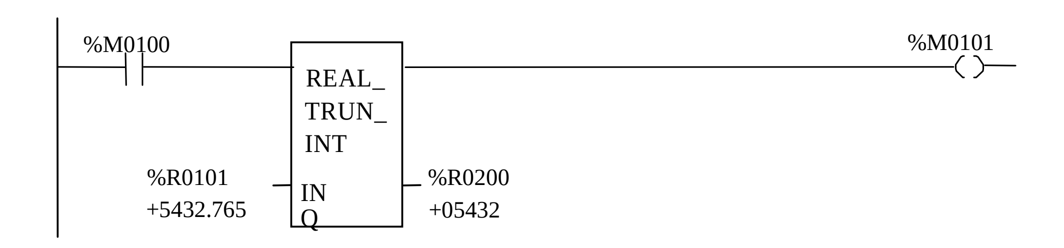
参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
IN								•	•	•	•	
ok	•											•
Q		o	o	o	o		o	•	•	•		

• 表示定流流过功能块的有效参考地址或位置。

o 仅对 REAL_TRUN_INT功能有效.

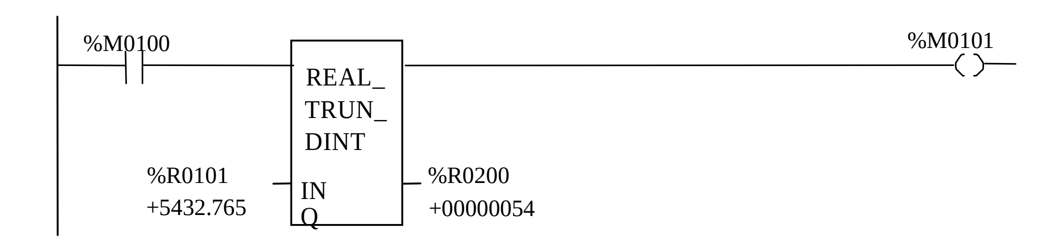
示例 1 –CPU352的REAL到INT型的舍位和输出线圈

在下例中，在%R0101中的值被舍位（舍弃十分位）并且结果位+05432被放置在%R0200中，如果其他35x, 36x, 或 37x CPU使用，在OK输出没有电流流通，所以不能编辑输出线圈。



示例 2 –CPU352的REAL到DINT型的舍位和输出线圈

在下例中，在%R0101中的值被舍位（舍弃十分位）并且结果位+0000005432被放置在%R0200中，如果使用CPU352，%M0101将被置ON，表示转换成功；如果其他35x, 36x, 或 37x CPU使用，在OK输出没有电流流通，所以不能编辑输出线圈。



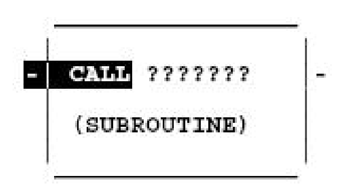
Chapter 12 第十二章 控制功能

这一章主要介绍控制功能模块，它们可以用来限制程序的执行和变动执行程序的 CPU 通道。有关 CPU扫描的内容可参考第2章第1节的“PLC扫描概述“。

功能	说明	页码
CALL	使程序的执行进入特定的子程序模块	12-2
DOIO	对输入或输出一特定范围的一次扫描立即服务。如果处于模块的任何给定存储单元都包含在DO I/O功能模块内，则该模块的所有输入或输出都将被服务。部分I/O模块将不进行更新。一般来讲，被扫描I/O的复制宁可存入内部存储器，也不放在真实的输入点上	12-3
SER	连续事件记录仪采集一系列的样本，该功能控制模块包含可由用户配置功能模块的执行、采样值类型和操作参数的功能	12-8
END	提供一个逻辑的暂时结束，该程序从第一回路执行到最后一个回路或END指令（以第一个遇到的END指令为准）。该END指令用于调试程序是有用的，但不能用于SFC编程（参考12-8页的注释）	12-23
MCR 与 MCRN	将以主令控制继电器编程。一个MCR将产生介于MCR与其后的不通电而执行的ENDMCR之间的所有回路。 Logicmaster 90-30/20/Micro 软件支持MCR功能模块两种方式，一种是非嵌套方式(MCR)，另一种是嵌套方式(MCRN)	12-24
ENDMCR 与 ENDMCRN	显示在正常通电状态下而被执行的后继逻辑。Logicmaster 90-30/20/Micro软件支持ENDMCR功能模块两种方式，一种是非嵌套方式(ENDMCR)，另一种是潜逃方式(ENDMCRN)	12-30
JUMP 与 JUMPN	将引起程序的执行进行逻辑跳转至一特定的存储单元（由一标记LABEL来显示，见后面）。Logicmaster 90-30/20/Micro软件支持JUMP功能模块两种方式，一种是非嵌套方式(JUMP)，另一种是嵌套方式(JUMPN)	12-31
LABEL 与 LABELN	JUMP指令的特定目标存储单元。Logicmaster 90-30/20/Micro 软件支持LABEL功能模块两种方式，一种是非嵌套方式(LABEL)，另一种是嵌套方式(LABELN)	12-33
COMMENT	将一个注释（回路注释）置入程序中。指令编程后，注释文本可通过缩放功能 “zooming “打印进入指令（用F10进行缩放）	12-34
SVCREQ	请求一个特别的PLC服务。（详见12-35页的服务请求列表）	12-35
PID	提供两种PID（比例/积分/微分）闭环控制算法 • 标准ISAPID算法(PIDISA) • 独立项算法 (PIDIND).	12-70

CALL

CALL功能用于调用指定子程序。



当CALL指令流过电流时，它将使扫描立即进入预先的子程序块并且执行它。当该子程序块执行完毕后，立即返回执行CALL指令下一个语句。

示例

本例中，当触点%I0006为ON，调用子程序块ROTATE.(注意当你调用某子程序块时，该子程序应该已经在块定义中定义过.)　光标定位至CALL指令处，按F10进入子程序。一旦子程序被调用，程序将执行该子程序块，执行完毕后执行CALL指令下一指令。本例中，在第二行调用子程序，因此执行完该子程序后，程序将返回到CALL指令下一行即第三行。

```
|
|%I0004                                                    %T0001
|—| |—————————————————————————————————————————————————————————( )—
|
|              _______________
|             |               |
|%I0006       | CALL ROTATE   |-
|—| |—————————| (SUBROUTINE)  |
|             |               |
|             |               |
|              _______________
|
|%I0003   %I0010                                           %Q0010
|—| |——+——| |—————————————————————————————————————————————————( )—
|      |
|%I0001 |
|—| |——+
|
```

注意

Micro PLCs 不提供子程序块; 因此CALL指令不适于Micro PLC.

DOIO

DO I/O (DOIO)功能模块用来处理程序运行时，为每次扫描更新输入与输出。除了常规I/O扫描外，DOIO功能模块还可用来对编程期间所选的I/O接口进行更新。 在常规的情况下，当PLC扫描程序中的输入扫描部分执行时，输入列表是可以被刷新的，并且直到下一次扫描的时候才会被再一次刷新。当PLC扫描程序中的逻辑执行时，输出列表是被刷新的，但是输出模块直到逻辑执行部分结束才会被刷新。由于DO I/O 功能模块，当一次扫描的逻辑执行部分运行时，输入列表和输出模块的更新是被强制执行的。 由于这一功能，用户可以比一般的PLC扫描更快地读到输入的改变和向输出模块写数据。关于PLC扫描的更多信息可参考第2章。
如果规定了输入参考地址，则功能模块将允许为程序逻辑获得输入的最新数值（写到输入列表中）。如果输出地址已经标定，则DO I/O 将根据存储在I/O存储器中的当前数值把输出模块更新。以整个I/O模块的增量来使I/O服务；如果必要的话，在功能执行过程中，PLC可调整给定地址。

输入模块的使用

DOIO 功能模块有4个输入参数和1个输出参数。当功能模块接通，并指定了输入给定地址时，从给定地址(ST)起始而在END指令处结束的输入点将被扫描。如果为ALT指定一个给定地址，则新输入值的复制被从存入以那一给定地址为起始的存储器中，且真实的输入点则不被更新。ALT 必须与被扫描的给定类型容量相同。对于ST与END采用数字量给定，则ALT页必须时数字量的。如果没有为ALT指定给定地址，则真实的输入列表将被更新。

输出模块的使用

当DOIO 功能模块接通，并指定了输出给定地址，则从给定地址(ST)起始至END结束的输出点将被写入输出模块。如果输出应从内部存储器写入输出模块，而不是写入%Q 或 %AQ存储器，则可为ALT指定起始给定地址。写入输出模块的范围由起始给定地址(ST)和结束给定地址(ECD)来规定。

功能模块的执行会一直进行，直到在所选范围内的所有输入值都已登记，或I/O模块中的所有输出都已服务。然后，程序执行则回到紧接DO I/O的下一个功能模块。

可选模块的使用

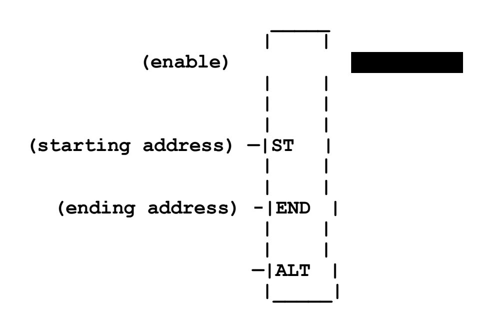
如果给定地址的范围包括一个可选模块（HSC,APM,etc），那么这个模块的所有的输入数据(%I and %AI) 或所有的输出数据 (%Q and %AQ)都会被扫描。 扫描可选模块时，ALT 的参数被忽略。同样的，如果希望把DOIO作为一个增强型的GCM模块(IC693CMM302)来使用，那么要满足下面注意事项中提到的要求。

注意

DOIO 功能只能在90以及更先进的CPU系统中作为增强型的GCM 模块(IC693CMM302) 使用

每当电源接通，功能模块便向右输出，除非：

- 并非所有指定类型的给定地址都存在于所选范围内
- CPU 不能正确处理由功能模块所产生的I/O暂存目录
- 所指定的范围包含那些于“I/O丢失”故障有关的I/O模块



参数

参数	说明
enable	当该功能被使能，一个有限的输入或输出扫描被执行
ST	ST 是一个输入或输出组的起始地址，或被服务的输入/输出点或字的起始地址
END	END是一个输入或输出组的结束地址，或被服务的输入/输出点或字的结束地址
ALT	对于输入扫描，ALT 将指定存储被扫描输入点/字数值的地址。对于输出扫描，ALT 指定从发送到I/O模块处获取输出点/字数值的地址。 对于331 或更先进的CPU, ALT 参数可影响DOIO功能块的执行速度(见下面的注释和本章中关于应用于331或更新的CPU增强型的DO I/O功能部分). **如果ALT功能没有被使用，这个输入将会被留下来不用;如果常数0作为ALT的参数, CPU会出现定时器超时的错误。**
ok	当输入或输出扫描正常完成后，OK输出便接通

注意

对于331型和更先进的CPU，一个增强型的DOIO 功能可以被使用。在增强型的DOIO模块中，ALT参数可以用来进入主机架中一个离散输入或输出的单模块插槽中。这个增强型的DOIO功能可以在80微妙内执行，而没有ALT参数的DOIO的执行需要236微妙。无错误检查执行用于防止指定地址重叠或模块类型不匹配。详见本章后面的“增强型DO I/O 功能” 部分。

有效存储器类型

Parameter	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
enable	•											
ST		•	•						•	•		
END		•	•						•	•		
ALT		•	•	•	•		•	•	•	•		•
ok	•											•

- 当模块通流时的有效指定地址或位置

输入示例 1

在下例中，当使能输入%M0001接通时，从%I0001 (在ST中) 到%I0064 (在END中)的给定地址将被扫描，并且%Q0001被接通。被扫描输入的复制值则存在从给定地址为%M0001 (在ALT中)到%M0064的内部存储器中。因为一个预备区域在ALT中是专用的，所以%I 输入列表是不会被DO_IO更新的。功能模块的这种方式可用来比较输入点的当前值与它们的原始值 (即它们在逻辑执行开始处的值)。

```
|
|          _____
|%M0001   |     |                                                   %Q0001
|--| |----|DO_IO|---------------------------------------------------( )-
|         |     |
|         |     |
|%I0001  -|ST   |
|         |     |
|         |     |
|%I0064  -|END  |
|         |     |
|         |     |
|%M0001  -|ALT  |
|         |_____|
|
```

输入示例 2

在下例中，当使能输入%M0001接通时，从%I0001 (在ST中) 到%I0064 (在END中)的给定地址将被扫描，并且%Q0001被接通。因为没有预备存储区域在ALT中是专用的，所以扫描的输入值被DO_IO用来更新从%I0001到%I0064输入列表。功能模块的这种方式在CPU扫描的逻辑执行中允许一次或多次扫描输入点。要注意到当ALT输入没有使用时，它将被留出来作为空白，如下所示。不要把“零”放到ALT输入中，因为这将会引起定时器出错。

```
|
|          _____
|%M0001   |     |                                                   %Q0001
|--| |----|DO_IO|---------------------------------------------------( )-
|         |     |
|         |     |
|%I0001  -|ST   |
|         |     |
|         |     |
|%I0064  -|END  |
|         |     |
|         |     |
|        -|ALT  |
|         |_____|
|
```

输出示例1

在下例中，当使能输入%M0001接通时，模拟量输出通道%AQ001 (在ST中) 到%AQ004 在(END中)的值被分别写到给定地址为%R0001 (在ALT中)到％R0004中，并且 %Q0001被接通。 因为%R0001预备区域在ALT是专用的，在%AQ001到%AQ004的数值则不写入模拟量输出模块.

```
|
|           _____
|%M0001    |     |                                                         %Q0001
|--| |-----|DO_IO|---------------------------------------------------------( )-
|          |     |
|          |     |
|%AQ001  - |ST   |
|          |     |
|          |     |
|%AQ004  - |END  |
|          |     |
|          |     |
|%R0001  --|ALT  |
|          |_____|
|
```

输出示例 2

在下例中，当使能输入%M0001接通时，给定地址%AQ001到 %AQ004的数值被适用的模拟量输出模块写到%AQ001到%AQ001的模拟量输出通道，并且%Q0001被接通。因为在ALT中没有预备存储区域，DO_IO要更新模拟量输出模块。要注意到当ALT输入没有使用时，它将被留出来作为空白，如下所示。不要把“零”放到ALT输入中，因为这将会引起定时器出错。

```
|
|           _____
|%M0001    |     |                                                         %Q0001
|--| |-----|DO_IO|---------------------------------------------------------( )-
|          |     |
|          |     |
|%AQ001  - |ST   |
|          |     |
|          |     |
|%AQ004  - |END  |
|          |     |
|          |     |
|        --|ALT  |
|          |_____|
|
```

用于331或更先进的CPU的增强型DO I/O功能

警告

如果增强型DO I/O用于一程序中，那么不能用低于4.01的Logicmaster 90-30/20 软件版本装载此程序。

DO I/O功能的增强型版本适用于331型的4.20版本或更高级版本以及更先进的CPU。DOIO功能的增强版只能用于单个离散输入或者8点、16点、32点的离散输出模块。

ALT参数表明模块所在的主机架中插槽。例如，在这种参数中，常数2表明模块在插槽2中。 ST和END参数设置了要执行的存储器范围。

注意

仅用增强型DOIO功能块执行的校验是一个对于目标模块情况的基本校验。

增强的DOIO功能只适用于放置在主机架中的模块。因此，ALT参数必须是一个5插槽机架的2和5之间或一个10插槽机架的2和10之间的数。

起始地址(ST)和结束地址(END)必须是%I或%Q。这些地址规定要被配置的模块的最初和最后的地址。例如，如果一个16点的输入模块被配置一个10插槽机架的插槽7中的%I0001 到%I0016 中，ST的参数必须是%I0001, END的参数必须是%I0016，ALT参数必须是10，如下所示：

```
|
|            _____
|%M0001     |     |                                                    %Q0001
|--| |------|DO_IO|----------------------------------------------------( )-
|           |     |
|           |     |
|%I0001 ---|ST    |
|           |     |
|           |     |
|%I0016 ---|END   |
|           |     |
|           |     |
|      7 ---|ALT  |
|           |_____|
|
```

下表把用于8点、16点、32点的离散输入/输出模块的一个普通的DOIO功能块与一个增强型DOIO功能块的执行时间仅需了比较：

模块	普通DOIO执行时间	增强DOIO执行时间
8点离散输入模块 8点离散输出模块	224微秒 208微秒	67微秒 48微秒
16点离散输入模块 16点离散输出模块	224微秒 211微秒	68微秒 47微秒
32点离散输入模块 32点离散输出模块	247微秒 226微秒	91微秒 50微秒

SER (连续事件记录仪)

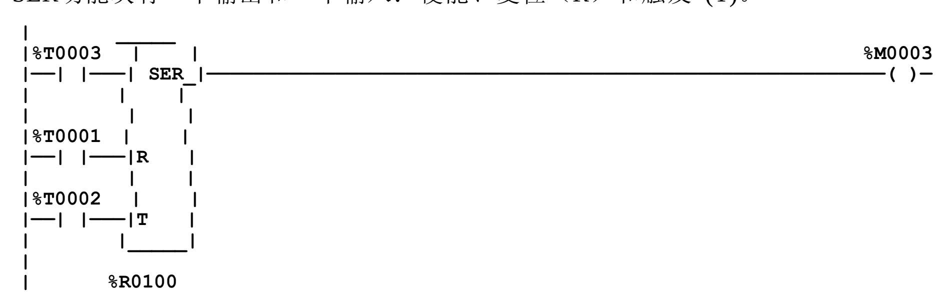
需要使用9.00或更高版本的固件的CPU 35x或36x，或者CPU37x

- SER (连续事件记录仪)功能块采集一系列的离散采样值（它只支持离散数据）。当使能输入接通时，对于每个采样值，一个SER功能块最多采集32个相邻的或非相邻的位。
- 每个SER最多采集1024个采样值，每个采样值32位。
- 如果SER功能块嵌套在一个周期执行的子程序中，它的采样频率以该程序的执行频率为基础。
- 只有触发器采样值是定时采样的。触发器采样值以BCD（最高分辨率是1秒）或POSIX式（最高分辨率是10毫秒）定时采样。 时间标记只被放置在触发点。SER每次记录作多支持一次时间标记。
- SER可以设置为预先、中途或事后触发模式。（见12-14页）
- SER操作可由一个功能控制块配置，用户可用Block Move (BLKMV)命令创建该块 。(见12-10页)
- 在SER执行时，一个随意指定的输入模块被扫描。这确保从指定模块采集到的数据尽可能是最新的。

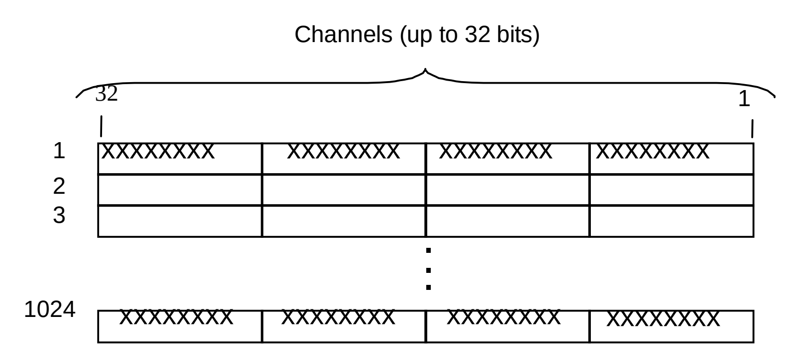
注意

不支持PLC跟PLC的同步。

SER功能块有一个输出和一个输入：使能、复位（R）和触发 (T)。



如下所示，8, 16, 24, or 32通道可被配置，每个通道支持一个离散点。最多支持1024个采样值。



参数

参数	说明
使能	当功能被激活，或复位输入断开时SER功能块从所有已配置通道采集一个采样值。
R	当复位输入接通时，SER功能被复位不管使能输入的状态，采样缓冲器、触发器采样偏移量、触发器定时、电流采样偏移量被清零，功能块保持复位状态直到复位输入端断电。在复位状态时，OK输出端断开。当复位端断电，继续采样。
T	如果选择触发器输入模式并且功能块被激活，当进行触发器输入时，SER转换为触发状态。触发器定时、触发器采样偏移量、数据采样值被记录。不管采样值的数目，触发器采样值将被记录。一旦触发，事件记录仪连续采样直到采样值数目达到要求，此时停止采样，直到复位输入端得电。如果触发模式被设置为全部缓冲器，触发信号被忽略。触发模式设置信息参见12-10页的“功能模块”。
起始地址	78个字的功能控制模块排列在这个地址开始。功能模块定义功能模块的执行、采样值配置和操作参数。详见12-10页的“功能控制模块”。
ok	当触发条件满足时(满足触发模式参数)，OK输出被激活，所有的采样完成。不管使能输入的状态，输出端持续得电，直到复位端得电。

有效存储器类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	const	none
使能	•											
控制块								•				
R	•											
T	•											
ok	•											•

• 当模块通流时的有效指定地址或位置。

功能控制模块

功能模块是一个78-字数组，它定义了SER的关于数据获取的信息和触发机制。在特定的程序里只有连续事件记录仪功能模块能够跟每一个功能命令块和数据块相联系。按照下列步骤设置SER功能块的参数：

1. 按照下表定义的方式为数组设置存储值。用户可以用块移动来对寄存器或寄存器列表中的数据进行初始化并且在激活SER功能前储存列表。
2. 把SER功能模块加入到你的梯形图。

注意

如果你需要x通道而x通道不是8，16, 24,又比32小,你必须选择一个比x大又是8得倍数的通道数，对于一个未使用的通道要填写一个无效通道说明。一个无效通道说明包括一个0xFFh 的区域选择、一个等同于未使用通道数的长度参数和一个“0”偏移量。

字	参数	说明
0 (起始地址)	状态	读取只能指示SER功能块当前状态的变量。附加的信息是在额外状态数据中被提供的(Word 1)。**注意:**如果在控制块中发现一个错误，状态将被设为6，OK输出被清零，并且没有任务执行。设定状态包括： 0 =复位 1 = 停止 2 = 激活 3 = 触发的 4 = 完成 5 = 超限错误 6 = 参数错误
1	状态附加数据	一个只读变量，它提供关于SER功能的附加信息。如何设置此参数见12-12页的“状态附加 数据状态。
2	触发器模式	定义了SER功能块转换为触发器状态的条件。正确的设置是： 0 = 触发器输入模式 1 = 全部缓冲器模式 在触发器输入模式中，如果功能块被激活，当触发器信号被激活时产生一个时间标记。采样一直持续，直到触发后采样数目达到要求为止。如果出现此情况，OK输出被激活。
		在全部缓冲器模式中，触发器信号被忽略。当功能模块被激活，采样一直持续，直到采样缓冲器被写满。此时OK输出被激活。采样数目这个参数设置缓冲器大小。
3	触发器定时格式	确定触发器如何被显示。用BCD显示，设置此参数为0，用POSIX显示，设置此参数为1(详见2-17页)
4—7	备用	4到7个字备用，设为0。

字	参数	说明
8	通道数目 (每个采样值的位)	指定数据的位的个数，这个数据在功能块执行时被采样并复制到采样缓冲器。有效的选择是8, 16, 24, 或32 位。任意未被使用的通道必须用无效通道说明配置。(见14—77字)例如，如果19个位是必须的，那么你必须配置24并且指定最后的5个是无效的。
9	采样值数目	指定采样值缓冲器大小。有效的选择是1到1024个采样值。(缓冲器的位的实际大小是通道数目的采样次数。)
10	触发后采样值数目	指定采样值数目，这些采样值是当触发条件为真后采集的。这个参数可设为0到采样值数目之间的一个数。此参数仅在触发器模式被设为0（触发器输入）时才有效。
11	模块插槽	指定一个输入模块主机架（0机架）的位置，这个模块在SER执行每次都被扫描。如果数值为0，没有模块被扫描。当输入模块被扫描，它的值被局部存储起来，为模块配置参考地址值无效。为了把数值从被扫描的输入模块储存到数据块采样缓冲器中，必须提供通道说明。如果模块不是当前的，或者在扫描时出错，数据将归零。此时，错误不会被记入错误列表；错误显示将保留到IO扫描器中。
12	数据块段选择器(存储器类型)	为数据块指定数据类型。例如，如果你要用%R 存储器，此参数置为08。这个参数有效的设定为： %R (08h), %AI (0Ah), %AQ (0Ch)。详见12-13页。
13	数据块偏移	指定数据块的起始指定地址。此参数以0为基础。例如，如果你想从%R0100开始，你就要将此参数设为99。要确定对于全部的数据块有足够的存储器。
14—77	通道说明	指定与特定通道相关的参考区域 (段选择，长度和偏移量) 。根据采样通道数目和数据长度可以有1到32个通道说明。在本节数据按照定义的顺序被回归。
	通道段选择/长度	作为一个十六进制数值被输入，这个字定义了段选择器和数据长度 (按位). MSB = 段选择器，LSB = 数据长度。数据长度对于连续采样值是有效的。 段选择器可以被设为任意的离散数据类型。：%I (46h), %Q (48h), %M (4Ch),%T (4Ah), %G (56h), %S (54h), %SA (4Eh), %SB (50h), %SC (52h), 无效的选择器 (FFh), 输入模块选择器 (00h)。
		长度参数范围是1—32, 但所有长度的总数必须不能超过通道个数参数。超过1的长度允许多个连续通道用一个单独的通道说明来配置。
	通道偏移量	作为一个十六进制数值输入，这个字为数据类型或在段选择器中指定的输入模块定义了位偏移量。这个偏移量以0为基础。此参数变化范围基于段选择器（数据类型和长度）。偏移量指示了数据列表内部或采样出入模块的位置。

状态附加数据状态

状态附加数据 (功能控制块中的字1) 为SER功能提供了附加状态信息。

数值	状态	说明
0	复位状态	复位输入得电，采样缓冲器、触发器采样值偏移量、触发器定时和当前采样值偏移量被清零。输出保持不得电。当复位端断开，转换为激活状态。状态附加数据无意义并将被清零。
1	非激活	处于复位状态和激活状态之间的状态。在这种状态下没有进城被执行。SER的输出保持没有通电。 当功能块收到使能电流时，转换为激活状态。
2	激活	使能输入已得电，但功能块没有被复位，出错或被触发。当功能块被激活时，每次执行一个采样值被记录下来。输出保持没有导通。触发条件（由触发模式参数设定）被监控，如果条件为真就转换为触发模式。如果“采样值数目”超出，状态附加数据将被设为0x01，否则为0x00。
3	触发	如果由触发模式指定的触发条件为真。附加采样要以触发模式和参数设定为基础。保持不导通。当所有的采样完成时，转换为完成状态。如果“采样值数目”超出，状态附加数据将被设为t 0x01，否则为0x00。
4	完成	所有的采样完成。输出得电。只允许转换为复位状态。如果“采样值数目”超出，状态附加数据将被设为t 0x01，否则为0x00。
5	超限错误	控制/数据块超过了存储器类型的限度。输出保持不导通。只允许转换为复位状态。状态附加数据无意义，并将被清零。
6	参数错误	在功能块或其他操作参数中有错误。输出保持不导通。只允许转换为复位状态。状态附加数据字包含控制块的偏移量，参数错误出现在这个控制块中。
7	状态错误	状态参数无效。输出保持不导通。只允许转换为复位状态。无效的状态值将被储存在控制块中的状态附加数据区。

SER数据格式

SER数据块包含采样缓冲器、采样偏移量和触发器信息。这个信息由CPU提供，用户只能从这个数据区读取。用户必须为数据块分配足够的寄存器空间。块格式如下：

字*	参数说明
0	当前采样值偏移量数字。指定区域存放了大部分的当前采样值。参数以0为基准。有效的范围是–1到1023。 采样值寄存器位置= (每个采样值的字节数) * (偏移量参数)/2 + (采样值缓冲器起始寄存器). **注意:** 数值一直无效，直到一次采样发生。当SER功能通过复位输入被复位时，这个数值被设为–1。
1	触发器采样值偏移量数量。当触发条件转换为真时，指定获得的采样值的存储区域。参数以0为基准。有效的范围是0到1023。 采样值寄存器位置= (每个采样值的字节数) * (偏移量参数)/2 + (采样值缓冲器起始寄存器)。 **注意:** 数值一直无效，直到一次采样发生。当SER功能通过复位输入被复位时，这个数值被设为0。
2 到 5	触发时间：根据PLC的时钟时间显示时间。在功能块内部触发器条件转换为真。时间值可以用BCD格式（默认）POSIX格式显示。此格式由控制模块中Th的触发时间格式参数决定。当复位输入激活时这个值初始化为。
6 到采样值缓冲器的限度	采样缓冲器。 保留数据采样值存储器的区域。当复位参数得电这个区域被设为0。采样缓冲器大小根据通道数量和采样值大小变化。当最后的区域被写入，采样缓冲器为循环缓冲器，后一个采样值将覆盖第一个寄存器。 采样缓存器限度=5 + ({[(# 被采样的值) * (# 被采样的通道/ 8)] +1} / 2

*从起始指定地址的偏移量由由功能块中的数据块段选择器（12字）和数据块偏移量（13字）指定。

SER 操作

在扫描时，如果SER被激活。它读取已配置的采样点并把它们放在环形列表中。在采样值数量被配置后，输出得电。输出的转换可以被用来记录时间，这个时间是最后一次采样的时间或开始附加采样的时间。 (见 “采样模式”)

在采样开始前SER功能块必须被复位（激活复位输入）。复位初始化数据块区域。如果功能块状态没有被复位，它将以数据块的当前状态来执行，会引起数据块中当前采样值偏移量出错或无效。

SER功能块中的控制块在功能块每次以复位、激活或触发状态执行时被扫描。如果用户在程序执行期间改变了一个控制块中的配置参数， 这个变化在与控制块相关联的SER功能块被扫描时起作用。如果发生一个错误，操作停止并且功能块处于一定的错误状态。

用户必须纠正错误，并复位功能块（激活复位输入端）进行再一次采样。

如果用户选择一个被扫描的输入模式，PLC不会校验模块是一个离散输入模块，或与模块相连续的通道说明，此模块具有有效长度和基于模块容量的偏移量。用户必须正确设置输入模块的采样。虽然多通道说明可以把一个输入模块作为对象，但模块仍只能在每个功能块执行时被扫描。

SER功能块可以放置在一般的用户逻辑程序中或一个周期子程序中。如果放置在用户逻辑程序中，扫描之间的时间间隔由扫描时间决定。这个扫描时间分辨率可根据任意特殊扫描功能的数目和类型改变。如果放置在一个中断子程序中，时间间隔可以设为1毫秒，分辨率将以1毫秒（带有很小的抖动）被高速的重复。

以1毫秒为周期的子程序中的某功能块的执行时间最多消耗CPU资源的50%。用户不可以在以1毫秒为周期的子程序中安排超过两个SER功能。

采样模式

SER采样模式由触发器模式（功能控制块的字2）和触发后采样值的数目（字10）参数来确定。用户将基于你所配置的参数来给出采样缓冲器的容量。

下表总结了如何确定采样模式。

模式	字2	字 10
预先触发	0	0
中途触发	0	从1 到 (采样值个数 – 1)
事后触发	0	等于采样值个数（由字9指定）
全部缓冲器	1	字10 和触发器出入信号被忽略

受控触发器采样

为了配置预先、中途和事后触发器采样模式，触发器模式（字2＝0）必须被选择。采样模式由后触发采样值的个数（字10）来控制。所有情况下，当使能信号为高电平时，采样开始。当触发器信号为高电平时，采样继续，直到由触发后采样个数参数指定的数字被采集到为止。当采样完成时，SER的OK输出信号为高电平。

如果在触发后采样个数条件被满足之前有超过已配置的采样个数（字9）被采集到，缓冲器“循环覆盖”，意味着SER返回到缓冲器起始的地址并覆盖最初的采样值。

当触发器第一次由关闭转换为打开时，触发时间被放置在一个已配置的区域。

预先触发

持续采集采样值直到触发被发现。

为了配置此模式，把字10设为0，当触发信号被激活时，采样停止并且产生一个时间标记（所有的采样值在触发前被采集）。

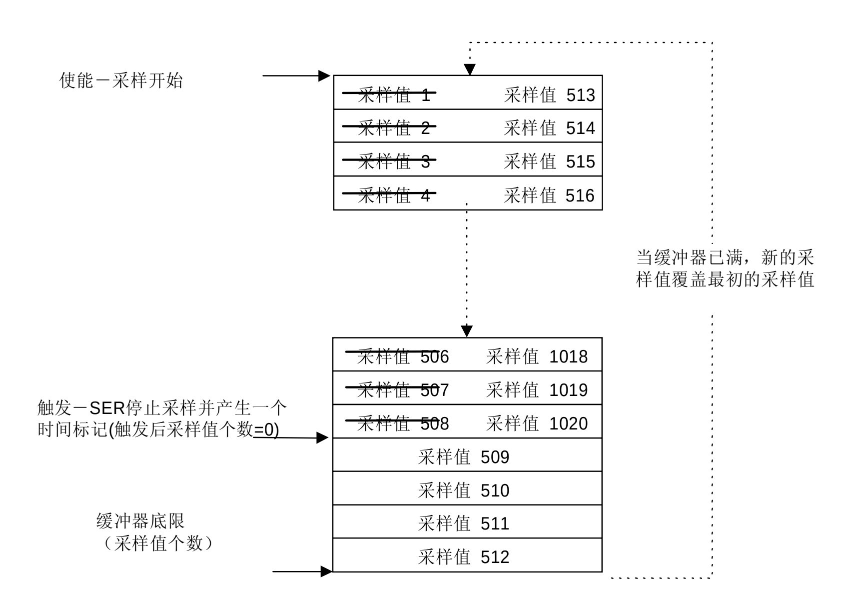


图 12-1. SER预先触发采样的例子 (以512个采样值为例)

中途触发

持续采集采样值直到达到触发后采样值个数。

为了配置此模式，把字10设为从1到（采样值个数－1）的数字，当触发信号被激活时，持续采样直到已配置个数被采集到。.
在下例中，触发后采样值个数为12。当采样完成时，缓冲器将保留500个预先触发采样值和12个事后触发采样值。

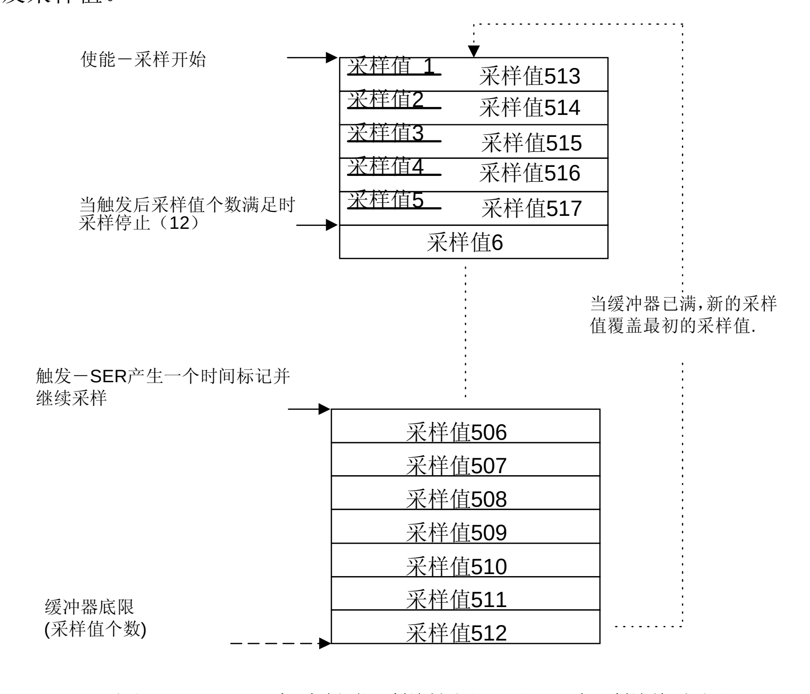


图 12-2. SER中途触发采样的例子(以512个采样值为例)

事后触发

持续采集采样值直到达到采样值个数。

为了配置这个模式，把字10设为等于采样值个数（字9）。当触发信号被激活，持续采样直到设置的个数已被采集到。
(注意：所有的采样值在触发后被采集。)

使能－采样开始

~~采样值 1~~ 采样值513
~~采样值2~~ 采样值514
~~采样值3~~ 采样值515

当缓冲器已满，新采样值覆盖最初值。

触发－SER产生时间标记并继续采样
(触发后采样个数 = 512, 等同于采样值个数参数)

~~采样值505~~ 采样值1017
采样值506
采样值507
采样值508
采样值509
采样值510
采样值511
采样值512

当触发后采样值个数满足，采样停止

缓冲器底限
（采样值个数）

图 12-3. SER事后触发例子(以512个采样值为例)

全部缓冲器(触发器不控制采样)

如果触发模式被设为1，触发后采样值个数参数（字10）被忽略，并且触发器输入信号不对功能块操作产生作用。当功能块被激活，采样继续直到采样值个数（字9）被采集到，填入采样缓冲器。当缓冲器已满，采样停止，产生一个触发时间标记，功能块OK输出端为高电平。

SER功能块触发器时间标记格式

BCD 格式		
数据模块字编号	**内容 (高字节/低字节)**	**推荐显示格式**
字 2	月/年	十六进制(MMYY)
字 3	小时/日期	十六进制(HHDD)
字 4	秒/分	十六进制(SSMM)
字 5	未用	全零

POSIX格式		
数据模块字编号.	**内容**	**推荐显示格式**
字2和 3	从1970年1月1日起秒的个数	凹痕
字4和 5	到下一秒的十亿分之一秒的个数	凹痕

实例

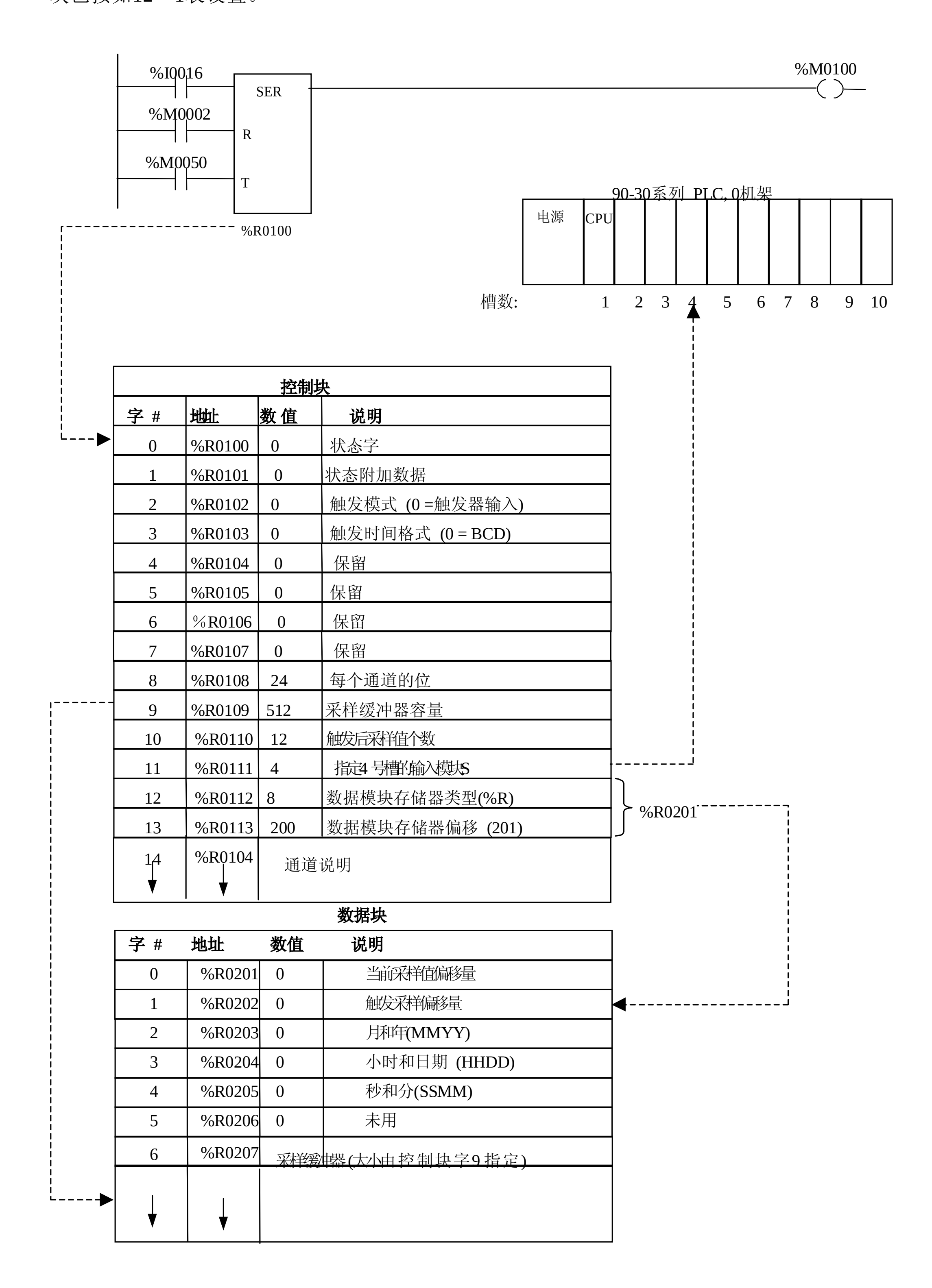
下面两个表格表明了1998年11月3日上午8:34:05.010在一个数据块中如何用BCD 和POSIX 格式显示，此数据块起始于%R0201 (字 0)。

1998年11月3日上午8:34:05.010 用BCD格式		
寄存器值	**参数**	**数值(十六进制)**
%R0203	月/年	1198
%R0204	小时/日期	0803
%R0205	秒/分	0534
%R0206	未用	0000

1998年11月3日上午8:34:05.010 用POSIX格式			
寄存器	**参数**	**数值(十进制l)**	**数值(十六进制)**
%R0203/R0204	秒	910,082,045	363EBFFD
%R0205/R0206	纳秒	010,000,000	00989680

SER示例

下列显示了梯形图指令、PLC存储器控制块和PLC中起作用的输入模块之间的相互关系。控制块已按如12－1表设置。



控制块

字 #	地址	数 值	说明
0	%R0100	0	状态字
1	%R0101	0	状态附加数据
2	%R0102	0	触发模式 (0 =触发器输入)
3	%R0103	0	触发时间格式 (0 = BCD)
4	%R0104	0	保留
5	%R0105	0	保留
6	%R0106	0	保留
7	%R0107	0	保留
8	%R0108	24	每个通道的位
9	%R0109	512	采样缓冲器容量
10	%R0110	12	触发后采样值个数
11	%R0111	4	指定4 号槽的输入模块S
12	%R0112	8	数据模块存储器类型(%R)
13	%R0113	200	数据模块存储器偏移 (201)
14 ↓	%R0104 ↓		通道说明

数据块

字 #	地址	数值	说明
0	%R0201	0	当前采样值偏移量
1	%R0202	0	触发采样偏移量
2	%R0203	0	月和年(MMYY)
3	%R0204	0	小时和日期 (HHDD)
4	%R0205	0	秒和分(SSMM)
5	%R0206	0	未用
6	%R0207		采样缓冲器(大小由控制块字9指定)
↓	↓		

功能控制块示例

在这个示例中，一个16点离散输入模块在0机架4号槽中，已被指定为采样的目标（在字11中）。它已被执行足够长的时间，以致572个（512＋60）采样值被采集。使能输入得电，但复位端和触发起输入没得电。

表 12-1. SER功能控制块示例

字	寄存器	参数	数值(十进制)	数值(十六进制)	说明
0	%R0100	状态	2	0002	功能块处于激活状态。意味着功能块正常执行，并当程序逻辑中每个功能块执行是采样。
1	%R0101	状态附加数据	1	0001	附加状态数据显示多于512个采样值已被采集，因此采样缓冲器至少已覆盖一次。
2	%R0102	触发模式	0	0000	基于触发器输入，时间记录仪被设置为触发器。
3	%R0103	触发器时间格式	0	0000	0=BCD
4	%R0104	保留	0	0000	保留参数通常设为0。
5	%R0105	保留	0	0000	
6	%R0106	保留	0	0000	
7	%R0107	保留	0	0000	
8	%R0108	通道数	24	0018	每个采样值由24位（3个字节）的数据组成。
9	%R0109	被采集的采样值数	512	0200	采样缓冲器容量是512个采样值。注意采样缓冲器等于512 x (24/8) = 1536 字节或768个字。(每个采样值是3个字节，由上面的字8指定。)
10	%R0110	触发后采样值数	12	000C	触发后将被采集的采样值个数是12。
11	%R0111	输入模块槽号	4	0004	当SER执行时，在0机架，4号槽中的输入模块被扫描，以致它的当前值被SER的采样所用。
12	%R0112	数据块端选择	8	0008	数据段是0x08 (%R).
13	%R0113	数据块偏移量	200	00C8	这个200的偏移量把数据块起始地址置于%R0201。偏移量是一个以0为基础的值，但寄存器列表起始于%R0001。因此，数据块起始点是%R0001 + 200 = %R0201。

接下页

字	寄存器	说明	数值(十进制)	数值(十六进制)	说明
通道说明		保持字包括通道说明。在此例中，六个通道说明已被定义。			
14	%R0114	设置选择长度	17921	4601	通道说明1：第一个通道说明以长度1，偏移量0选择%I 段。这个为通道1选择%I0001。
15	%R0115	偏移量	0	0000	
16	%R0116	设置选择长度	-253	FF03	通道说明2：第二个通道说明以长度3，偏移量0选择了空选择器。空选择器导致2－4通道被忽略或“跳过”。这些通道将始终包括一个0的采样值。
17	%R0117	偏移量	0	0000	
18	%R0118	设置选择长度	3	0003	通道说明3：第三个通道说明以长度3和偏移量12选择了输入模块。输入模块选择器引起从输入模块采样。这个通道说明为5－7通道在输入模块中的13、14和15点选择数值。
19	%R0119	偏移量	12	0012	
20	%R0120	设置选择长度	18434	4802	通道说明4：第四个通道说明以长度2和偏移量8选择了%Q段。这个为通道8和9选择了%Q0009和 %Q0010。.
21	%R0121	偏移量	8	0008	
22	%R0122	设置选择长度	8	0008	通道说明5：第五个通道说明为另一个输入模块选择器。它有长度8和偏移量0。这导致输入模块中1到8点的数值被放到通道10 – 17。
23	%R0123	偏移量	0	0000	
24	%R0124	设置选择长度	-249	FF07	通道说明6：第六个通道说明是另一个空选择器。它有长度7和偏移量0。这个空通道说明将使通道18 – 24被0填满。为了把采样缓冲器加长为24位（由通道个数参数指定），这个最后通道说明是必须的。因为所有的24通道被配置，所以不再需要通道说明。
25	%R0125	偏移量	0	0000	

上例的通道组态

32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
U	U	U	U	U	U	U	U	N	N	N	N	N	N	N	C8	C7	C6	C5	C4	C3	C2	C1	%Q 10	%Q 09	C15	C14	C13	N	N	N	%I 01

U =未用, N = 空l, C 表明配置输入模块的通道个数的前缀 (f例如, C0 = 输入点 1, C15 = 输入点 16)

实例采样内容

表 12-2 总结了包含在一个单一采样值的数值，这个采样值以采样控制块的通道说明为基础。这个基于实例屏幕获取在接下来的页中显示。注意在这个示例中，1－16位包含在%R00207 ，17 – 24 位是%R00208的一部分。

表 12-2. SER示例采样内容

通道数	通道内容	数值
1	%I0001	1
2 - 4	0	000
5	输入模块点13	1
6	输入模块点14	1
7	输入模块点15	1
8	%Q0009	0
9	%Q0010	0
10 - 17	输入模块点1 - 8	100100010
18 - 24	0	0000000

控制块示例的数据块

表12-3 列出了从12－19页给出的示例控制块得出的数据块格式。注意：它开始于寄存器201，被控制块中的段偏移量参数（字12和13）描述。

表12-3. SER控制块示例的数据块

偏移量	寄存器	参数说明	数值(十进制)	Value (十六进制)
0	%R0201	当前采样值偏移量	59	003B
1	202	触发采样值偏移量	0	0000
2 - 5	203 – 206	触发时间 (BCD)	0 0 0 0	0000 0000 0000 0000
6 - 768	207 – 975	采样缓冲器	采样数据	采样数据

当前采样值偏移量是59，意味着第59个采样值是最后一个被置于采样缓冲器中的采样值。由于每个采样值的3个字节，当前偏移量实际在59 * 3 = 177 字节或第89个寄存器的高字节。因为触发条件没有满足，触发器采样值和触发时间是0，输出没有没设置。采样缓冲器包含512个采样值，在这里，59是最新的采样值，60是最旧的采样值。

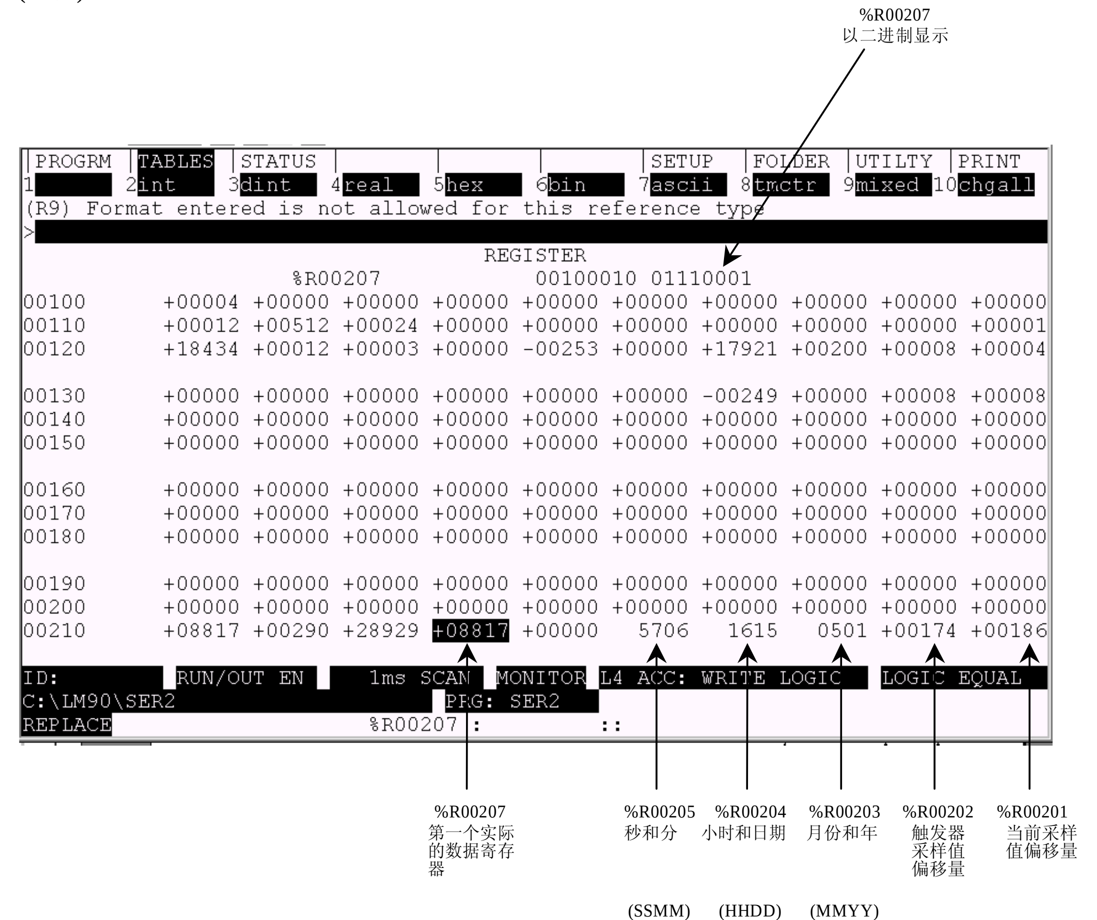
数据获取示例

检查获取的数据

下面的画面是触发后抓拍下来的。控制块起始于%R00100,数据区起始于 %R00201。光标放置在%R00207处，靠近屏幕的顶部和底部都有注释。

%R00207是数据块的第一个寄存器，此数据块实际上保存有标准的输入数据。注意，它的整型数据（－21855）在这种情形下没有意义。但是，如果把光标房子%R00207处, 它的数值将在屏幕顶部附近以二进制的形式显示。使用这种二进制格式，用和可以确定已在控制块的通道说明部分配置好的位的状态。

从寄存器%R00203到 %R00205给出了时间和数据，以24小时制的格式， 2001年5月15日，16:06 (57秒) 。



END

END功能块提供一逻辑暂时结束的功能。程序将从第一行执行到最后一行，或执行到第一次遇到END指令处。

END功能块将无条件中断程序的执行。在回路中，END功能块之后不能有其他程序。没有超过END功能的逻辑被执行，而控制将转到下次扫描的程序起始处。注意，在END标记后面的回路中，输入将出现打开和关闭，但输出不会被更新。尽管是一个普通的情形，但如果不显示一个在执行回路前的END标记，它将会出现问题。

END功能块由于它可以隔离一个逻辑段，因此对调试程序是有用的。通过中止任何逻辑的执行来完成这一功能。

Logicmaster 编程软件将默认在逻辑的最后一个回路后提供一个[END OF PROGRAM LOGIC]指示器，来显示程序执行的结束。如果没有END功能块在逻辑中编程，则使用该指示器。.

一 [END]

示例

在下例中，因为END指令的存在，包含有触点%I0222 、线圈%Q0017和其后的任意回路的一个回路将不被执行。

```
 %M0100                                          %M0101
--| |--------------------------------------------( )--

--[  END  ]

 %I0222                                          %Q0017
--| |--------------------------------------------( )--
```

注意

把END功能块置于SFC逻辑中或由SFC调用的逻辑中，会在版本7或更新的CPU中产生一个“从SFC操作中执行END功能”的故障。
(在版本7以前的CPU中，将不能正常工作，但没有故障产生。) 关于故障的详情，请参考第3章第2节中的“系统设置不匹配”部分。

MCRN/MCR

MCR和MCRN的综述

一个主令控制继电器 (MCR/MCRN)功能要与其相应的中止主路控制继电器(ENDMCR/ENDMCRN)功能结合使用。两个功能必须有相同的名字。MCR/MCRN在它和电源母线之间必须有一个使能触点。在一个已动作的MCR/MCRN和与其相应的ENDMCR/ENDMCRN功能之间的所有回路,即使线圈不得电仍照样执行。与MCR/MCRN 相关的ENDMCR/ENDMCRN功能用来恢复程序的正常运行。不想JUMP指令， MCR/MCRN仅在前进方向出现。ENDMCR/MCRN指令必须按程序在与其相应的MCR/MCRN指令后出现。

下列控制器由一个动作的MCR/MCRN来强制控制：

- 定时器不会增加或减少。任何TMR类型的定时器被复位（累加器被设为0）。对于一个ONDTR定时器，当MCR/MCRN被激活时，累加器停留在当前值。
- 电流不会出现在任何指令中。正向输出断开，反向输出闭合。
- 指令不更新它们的输出。例如，一个ADD指令在它的输出寄存器中不产生一个当前的和数，一个Move指令将不会把它的当前输入值复制到它的输出端，一个Shift寄存器不会转换数据，等等。 当MCR/MCRN被激活时，这些输出寄存器中的数值将停留在当前值上。

注意

当一个MCR/MCRN得电，它所控制的逻辑被执行，触点状态被显示，但没有输出得电。如果你没有意识到一个MCR/MCRN正在控制被观察的逻辑，这将导致一个故障。为了显示在MCR/MCRN控制下的梯形图逻辑的范围，Logicmaster将在梯形图逻辑屏幕上显示一个双电源母校。这个双电源母线不管MCR/MCRN 是否被激活都会显示。

Logicmaster 90-30/20/软件提供两种主令控制继电器功能方式，老版本的非嵌套形式（MCR）和新版本的嵌套形式（MCRN）。

CPU兼容性

CPU Type	Supported Form
CPU311 – CPU341, 版本 1	只用非嵌套形式(MCR)
CPU311 – CPU341, 版本2 和更新的	只用嵌套形式(MCRN)
35x, 36x, and 37x 系列 CPUs	只用嵌套形式(MCRN)

MCRN 可能存在的兼容性问题

当把一个CPU340 或CPU341程序转换为在35x/36x/37x系列CPU中运行时，在Logicmaster 90有可能出现一个“特征不支持”错误（“超越嵌套级别”）。如果有超过8级MCRN嵌套在原始程序中使用，当转换程序被储存在35x/36x/37x CPU 中时， 这个错误将会出现。

在CPU340/341中，MCRN指令是实际功能块指令，这意味着它们在CPU固件中被执行，不会被内置的布尔协处理器（BCP）激活。 对于此功能块，嵌套的极限是256。这个限制比你将会经常用到的要高很多级。当35x/36x/37x系列 CPU被使用时， MCRN指令被移到BCP中，以便改进CPU性能（ 功能块指令执行速度比BCP对应部分慢）。此时，嵌套级别和性能就要有一个权衡，在35x/36x/37x CPUs 中的BCP3 执行8级嵌套，这通常比用户需要的多。因此，当程序转换完成时，Logicmaster 90执行8级嵌套，并且，如果有超过8级的嵌套被使用，会出现一个 “超越嵌套级别” 信息。

因此，如果你在CPU340/341程序中使用超过8级的MCR嵌套级别，就需要在一个35x/36x/37x CPU 中做一个修改。你可以考虑使用JUMP语句代替。

嵌套MCRN

MCRN功能可以置于程序中的任何位置，只要它考虑同其他MCRN的关系而被适当的嵌套，并且，不会出现在任意非嵌套MCR或非嵌套JUMP的范围内。

如果成对的N/ENDMCRN在其他成对的N/ENDMCRN内嵌套，它必须完全包含在其他的那一对中。最多允许8级嵌套。见12-28页的示例。

与其相应的单一ENDMCRN，可以有多重MCRN功能 (除了如下列注释的35x/36x/37x 系列CPUs)。 每个MCRN和ENDMCRN必须有同样的名字。这与嵌套的JUMP相类似，在这你可以对多个JUMP使用同一个LABEL。JUMP和MCR的对比，参见下面的“MCR和JUMP之间的差异”部分。

注意

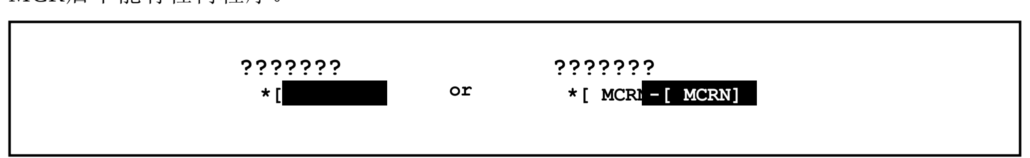
35x, 36x and 37x系列CPUs ，对于每个ENDMCRN只能用一个MCRN。

MCR 操作

对于每个ENDMCR 指令只能有一个MCR指令。非嵌套MCR和ENDMCR的范围不能重叠，或包含其他任何MCR/ENDMCR指令对或JUMP/LABEL指令对的范围。非嵌套MCR不能在任何的JUMP/LABEL指令对作用域内。

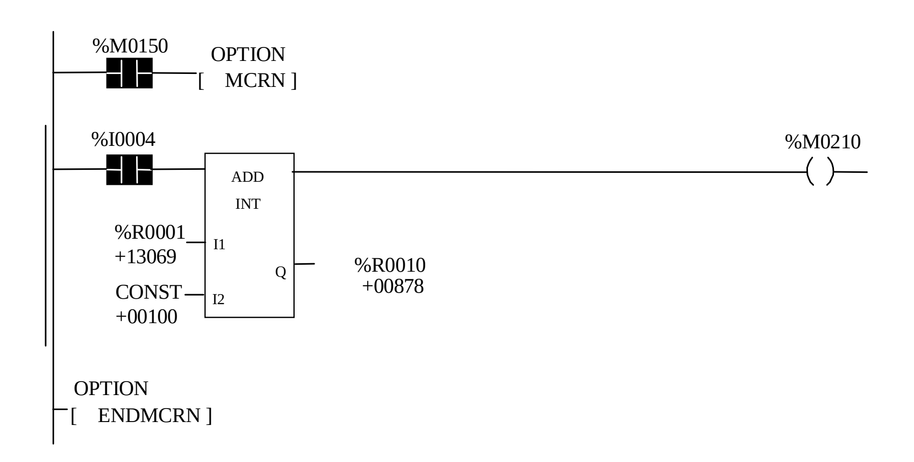
参数

两种MCR功能有同样的参数。它们都具有一个使能布尔型输入EN和一个识别MCR的名称。这个名称同样用在ENDMCR指令中。MCR和MCRN功能都不能有任何输出；一个回路中在MCR后不能有任何程序。

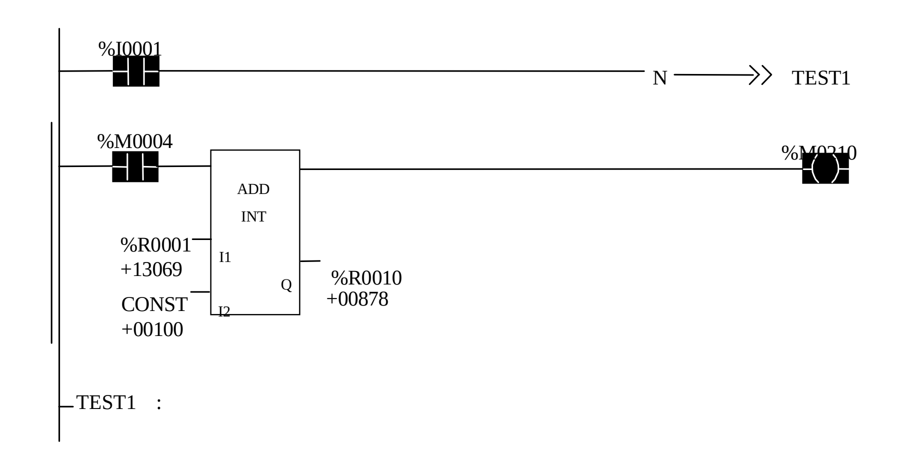


MCR/MCRN和JUMP的区别

对于一个MCR功能块，在MCR范围内的功能块在**没有得电**的情况下可执行，并且线圈是断开的。在下例中，当 %M0150 得电时，MCRN启动。当MCRN启动时，即使%I004得电，ADD功能块没有得电也能执行。(即它不把100加到%R0001)。且%M0210没有得电。触点（如%I0004）的状态和用在输入上的寄存器（如%R0001）的数值会在Logicmaster屏幕上更新；但在MCRN执行时，在MCRN控制下的寄存器的输出，象%R0010将停留在它们的当前值。



对于JUMP功能，JUMP和LABEL之间的任意功能块都**不能**执行，并且线圈**不受影响**。在下例中，当%I0001处于ON时，名为TEST1的JUMP被激活。既然JUMP和LABEL之间的逻辑被跳过， %M0210是不受影响的（即：如果它是ON则保持为ON，如果它是OFF则保持为OFF）。触点（如%M0004）的状态和用在输入上的寄存器（如%R0001）的数值将会在Logicmaster屏幕上更新。但在JUMP执行时，在JUMP控制下的寄存器的输出，象%R0010将停留在它们的当前值。



示例 1

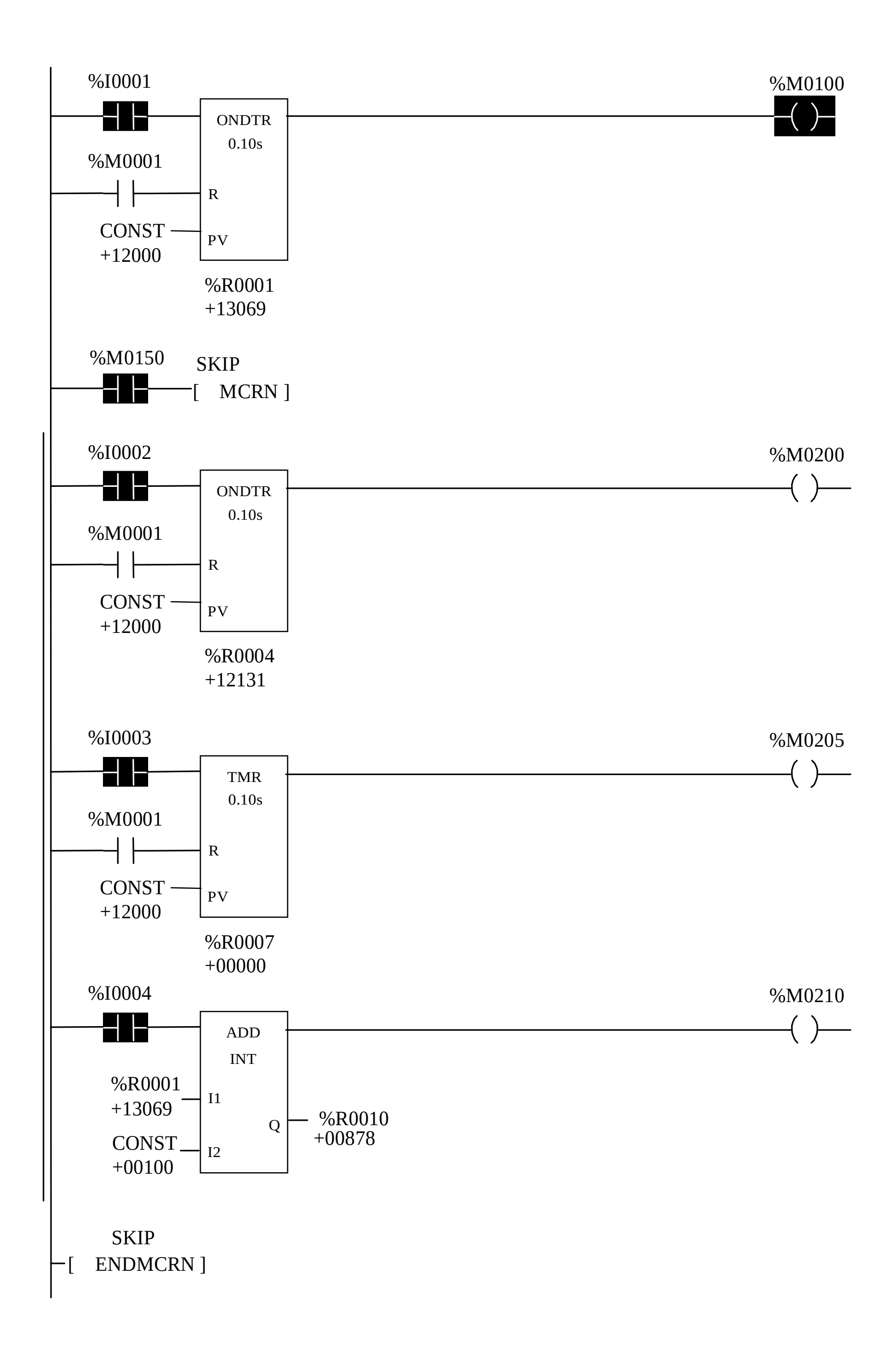
下例中，一个名为“第二”的MCRNQ嵌套在名为“第一”的MCRN中。只要%I0002允许电流进入MCRN模块，在电流没到线圈时，程序将持续执行直到到达与之相关的ENDMCRN。如果 %I0001和 %I0003 处于ON， %Q0001断开且%Q0003 保持 ON.

为了对发现并修理梯形图程序的故障有所帮助，一个双电源母线确定在MCR控制范围内的逻辑。

```
 |
 |%I0002   FIRST
 |——| |———[ MCRN ]
 |
||
||
||%I0004   SECOND
||——| |———[ MCRN ]
||
||
||
||%I0001                                                                    %Q0001
||——| |——————————————————————————————————————————————————————————————————————( )—
||
||
||
||%I0003                                                                    %Q0003
||——| |——————————————————————————————————————————————————————————————————————(S)—
||
||
||   SECOND
|+[   ENDMCRN   ]
||
||
||
 |    FIRST
 +[   ENDMCRN   ]
 |
```

示例 2

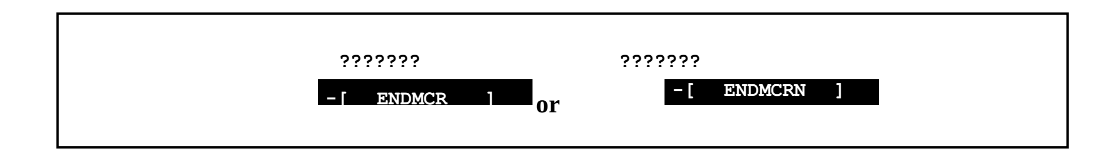
在下例中，第一行回路正常运行。名为SKIP的MCRN控制了其余的回路，这些回路有双电源母线来标明。在由MCRN控制的第一行中，ONDTR定时器的累加值(%R0004) 停住不动，即使它达到它的预设值，它的输出(%M0200)也不会被激活。接下来的回路中，TMR已被MCRN复位。它的累加值 (%R0007) 保持为0且输出(%M02025)不被激活。接下来的回路中，ADD指令的输出停止(它的输出%R0010不是输入的和) ，且它的线圈(%M0210) 不得电。注意，触点的状态和输入寄存器（如在ADD指令I1输入中的%R0001）在MCRN控制域内被刷屏显示。



ENDMCRN/ENDMCR

终止主令控制继电器ENDMCR/ENDMCRN功能用以在一个MCR/MCRN 功能之后继续程序的正常运行。当与ENDMCR相关的MCR动作时，ENDMCR将导致程序的执行以正常的流向继续进行。当与ENDMCR相关的MCR未起作用时，ENDMCR将不起作用。

Logicmaster 90-30/20/软件支持ENDMCR功能的两种形式，即非嵌套形式和嵌套形式。 非嵌套形式ENDMCR必须与非嵌套MCR功能的MCR一同使用。嵌套形式的ENVMCR必须与嵌套MCR功能的MCRN一同使用。 ENDMCR功能有一个反向布尔输入EN。指令使能必须由电源母线提供，其执行是无条件的。ENDMCR功能也有一个识别ENDMCR且与相应的MCR相关的名称。ENDMCR功能没有输出，在回路中ENDMCR指令之前或之后可无其他指令。



示例

下例中，一个ENDMCR指令变编程以结束名为“CLEAR”的MCR。

非嵌套ENDMCR的示例

```
|
|    CLEAR
|
|-[   ENDMCR    ]
|
```

嵌套ENDMCR的示例

```
|
|    CLEAR
|
|-[   ENDMCRN   ]
|
```

JUMP

利用JUMP指令以跳过一部分程序逻辑。程序的执行将在规定的LABEL处再继续进行。当JUMP起作用时，其范围内所有的线圈均将脱离其以前的状态。这包括一些与定时器、计数器、锁存器和继电器相关的线圈。

Logicmaster 90-30/20/软件支持JUMP指令的两种形式，非嵌套形式和嵌套形式。自用于CPU311-CPU341 CPU的版本1固件发行，非嵌套形式已被使用。且有形式
————————>>LABEL01,这里LABEL01为相应非嵌套LABEL指令的名称。

对于非嵌套JUMP，每一个LABEL指令只有一个单一的JUMP指令相对应。JUMP既可以是前向指令，也可以是后向指令。

非嵌套JUMP与LABEL的范围不能与任何JUMP/LABEL指令对或MCR/ENDMCR指令对的范围重叠。非嵌套JUMP及其相应的LABEL不能在任何其它JUMP/LABEL对或任何MCR/ENDMCR对的范围内。 除此之外，MCR/ENDMCR指令对或其他的JUMP/LABEL指令对不能在非嵌套JUMP/LABEL指令对的范围内。

注意

> JUMP指令的非嵌套形式是在90－30系列PLC版本1中唯一可用的JUMP指令。嵌套JUMP功能对所用的新应用均能使用（并推荐使用）
>
> 请注意 35x/36x/37x系列CPU只支持嵌套JUMP。

JUMP指令的嵌套形式有———N——>>LABEL01，这里LABEL01 是JUMP和相应嵌套LABEL指令的名称。对于Logicmaster 90-30/20/软件和PLC硬件的版本2和更新的版本，嵌套JUMP都适用。

嵌套JUMP指令可放在程序的任何地方，只要它不出现在任何非嵌套MCR或非嵌套JUMP范围内即可。

可以多个嵌套JUMP指令对应一个单一嵌套LABEL，嵌套JUMP可以是向前也可以是向后的JUMP指令。

两种形式的JUMP指令总是放在当前回路行的第九列和第十列，在回路中JUMP指令后可以没有其他指令。电流可直接从指令跳转到带名称标记的回路。

非嵌套JUMP:

——>> ???????

嵌套JUMP:

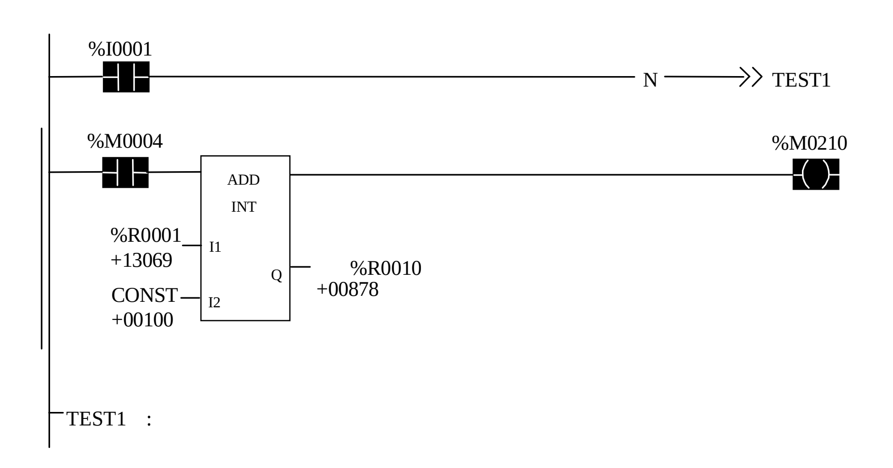
N——>> ???????

警告

为了避免产生向后JUMP指令的无终端环路，一个向后的JUMP指令必须包含使其变成有条件的一个通道。

示例

在下例中，只要触点%I0001闭合，名为TEST1的JUMP被激活，且电流跳到TEST1 LABEL前面。因为在JUMP和LABEL之间的逻辑被跳过，所以%M0210不起作用(即如果它处于ON则保持ON，如果它处于OFF则保持OFF)。触点（如%M0004）的状态和用于出入的寄存器（如%R0001）中的数值 将在Logicmaster 屏幕上更新。 但当JUMP执行时，在JUMP控制下的用于输出的寄存器（如%R0010）将停留在它们的当前值上。注意在JUMP和它的LABEL之间的部分逻辑中双电源母线的使用。.



LABEL

LABEL指令功能在于指定一个JUMP的目标。利用LABEL指令以使在一个JUMP指令后继续程序的正常执行。

在程序中，只能有一个具有特定标记名称的LABEL。没有匹配的JUMP/LABEL对，也可编程并存入PLC中，但不能执行。Logicmaster 90-30/20/软件支持两种形式的LABEL功能，非嵌套式和嵌套式。例如，非嵌套式LABEL01（————————>>LABEL01）与非嵌套式JUMP功能一同使用; 嵌套式LABEL01 (———N———>>LABEL01)必须同嵌套式JUMP一起使用。

LABEL 指令既没有输入也没有输出。在一个回路中的LABEL指令前或后都可以没有其他指令。

非嵌套式LABEL:

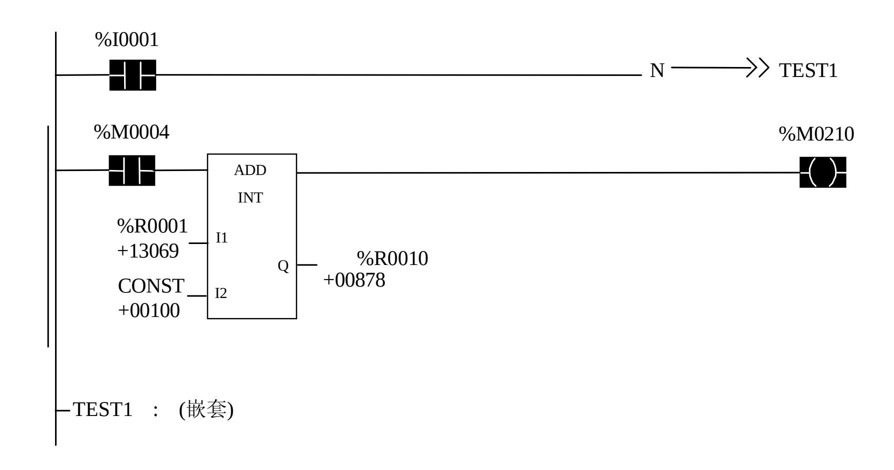
???????:

嵌套式LABEL:

???????:

示例

在下例中，当JUMP TEST1被激活，扫描跳到TEST1前：嵌套式LABEL，这意味着在JUMP和LABEL之间的回路不被扫描。



COMMENT

COMMENT可以用来加入说明、注释和版本信息等到你的梯形图程序中。推荐使用COMMENT，因为它们为那些将来必须解决检修故障或更新系统的人提供了有效的信息。既然人的记忆是有缺点的，因此COMMENT对于那些即使是梯形图程序的编写者来说，也是很有价值的参考。

注意

为了节省PLC内存注解（注释、绰号和描述）不被写入PLC。因此，为了看到这些注解，你必须在你的电脑中有一个最初程序文件夹（包括注解）的副本。这样，当你把你的电脑同你的PLC相连时，同那些注解的联系就会自动的由你的编程软件生成。

创建一个标准的注释

一个注释可有多至2048个文本字符。在Logicmaster中，它在梯形图逻辑中按如下方式表达：

```
(*  COMMENT  *)
```

创建一个注释t

1. 创建一个新行。一个 COMMENT 行不能有任何其他逻辑在其指令中。
2. 插入 COMMENT, 它将出现在指令控制组中。
3. 通过按空格键接受行。.
4. 将光标移动到(* COMMENT *) 指令处，创建且按下Zoom键(F10)来进入注释编辑界面。
5. 在你的注释文本中打字。注意，在注释编辑器中行不能自动换行，你必须在一行的末尾按回车键来开始在下一行打字。
6. 结束时，按退出键退出注释编辑器并保存注释。一旦创建， COMMENT文本可以通过把光标移到(* COMMENT *) 处并选择Zoom (F10)来读取或编辑。注释也可以通过Logicmaster的打印菜单打印。

在Logicmaster 打印输出中创建一个长注释来使用

在Logicmaster中长文本可以包含在使用一个注解文本文件的打印输出中。

1. 创建注释（详见前面的注释创建部分）。
   - A. 把注释文本送入文本从其他文件处应该开始的点。
   - B. 在一个行中，输入\I (or \i)，驱动器字母后跟一个冒号 和一个反斜杠，子目录或文件夹，一个反斜杠和一个文件名，如下例：

     ```
     \I d:\text\commnt1
     ```

     (如果文件存放在同程序文件夹相同的驱动器中，不需要驱动器指示。)
   - C. 按退出键退出注释编辑器并保存注释文本。
2. 打开一个文本处理器并创建一个文本文件。
3. 以.txt格式保存文本文件，给它一个以注释命名的文件名，把它保存在驱动器中一个注释专用的路径中。

SVCREQ

服务请求指令是一个普通作用指令，能完成多种特殊指令（服务），这些服务不能由单个功能块完成。使用服务请求指令（SVCREQ）函数需要如下的特殊PLC服务：

表 12-4. 服务请求功能

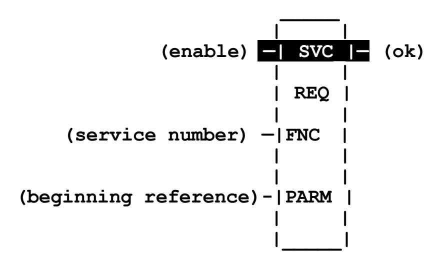
功能	说明
1	改变/读取恒扫描定时器
2	读窗口值
3	改变编程器通讯窗口模式和时间值
4	改变系统通讯窗口模式和时间值
6	改变/读取校验和任务状态和字数
7	改变/读取日历时钟
8	复位看门狗记时器
9	从扫描的开始读扫描时间
10	读文件夹名
11	读 PLC ID.
12	读PLC运行状态
13	关断PLC.
14	清故障表
15	读最后进入故障表的故障
16	读时钟的流逝时间
18	读I/O ，不考虑状态
23	读主检验和
24	复位精确的模板
26/30	询问 I/O.
29	读取失去电源时间
45	跳越下个输出和输入扫描 (悬挂 I/O.)
46	高速底板状态访问
48	致命故障自动复位后重新启动
49	自动复位统计表

SVC REQ 概述

SVCREQ函数有3个输入参数和一个输出参数。当SVCREQ收到电源流程时，要求PLC完成功能FNC的指示。功能的参数从PARM提到的参考基准开始。除非一个不正确的功能数，不正确的参数，或超出了指定的参考范围，SVCREQ 功能将流过电流流程。在后面将讲述其它引起故障的原因。

给PARM的参考基准可以是任意类型的字存储器 (%R, %AI, or %AQ). 这个参考基准是一组的开始，组成了功能的“参数块”。连续的16位位置存其它参数。要求的参考基准的总数取决于SVCREQ功能使用的类型。

参数块能被用于功能的输入和函数执行后的输出数据位置。因此，函数反馈的数据可在PARM指定相同存储单元处存取。



参数

参数	说明
enable	当使能接通时，即开始执行服务请求
FNC	每种服务请求有一个唯一的功能数，必须在FNC输入编译。FNC可能包含一个恒定值或一个参考地址（其包含了请求的服务功能数）。
PARM	PARM 包含请求服务的参数块的初始参考地址。
ok	当函数正确无误执行后，输出OK信号

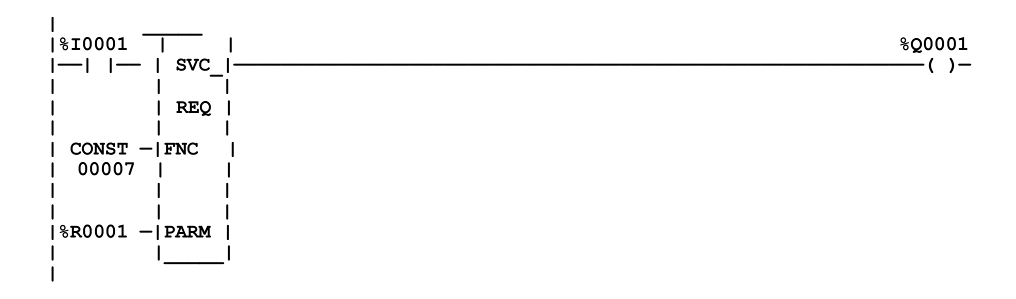
有效存储类型

参数	flow	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	常数	无
enable	•											
FNC		•	•	•	•		•	•	•	•	•	
PARM		•	•	•	•		•	•	•	•		
ok	•											•

• 使函数模块流通时的有效参考地址或位置。

示例

下例中，当使能接点 %I0001为ON, SVCREQ函数参数是 7,指定给FNC输入，执行． 功能的参数块从 %R0001 (PARM内指定)开始． 如果操作成功，输出线圈 %Q0001为 ON.



SVCREQ #1: 改变/读取恒定扫描定时器

90-30 CPU 8.0版以后，使用 SVCREQ函数#1:

- 不允许 **恒定扫描**模式.
- 允许 **恒定扫描** 模式和使用旧定时器值.
- 允 许 **恒定扫描** 模式和使用新定时器值.
- 只 设 定 新 的 定 时 器 值 .
- 读取 **恒定扫描** 模式 状态和定时器值.

注意

这个手册中讲的CPU，服务请求1 只被90-30CPU支持，8.0版以后。

参数块有2个字长.

为了不允许**恒定扫描** 模式, SVCREQ函数#1 内输入这个参数块:

参数	位置
0	地址
ignored	地址 + 1

为了允许**恒定扫描** 模式， SVCREQ函数#1 内输入这个参数块:

参数	位置
1	地址
0 or timer value	地址+ 1

注意

如果定时器想使用新值，在第2个字内输入其值。如果定时器值想不变，在第2个字内输入0。如果定时器值总已经不存在，输入0将引起功能的OK输出变为OFF。

为了改变定时器值，而不改变扫描模式的选定状态，SVCREQ 功能#1 内输入这个参数块:

参数	位置
2	地址
新定时器值	地址+ 1

为了读取现在定时器的状态和值，而不改变任何一个， SVCREQ函数#1 内输入这个参数块:

参数	位置
3	地址
忽略	地址+ 1

注意

在按照上面的参数块使用SVCREQ函数#1后，8版和更高版本的CPU在正常扫描时反馈0值，恒定扫描时反馈1值。不要同如下的输入值相混淆了。

将成功执行, 除非:

1．输入了一个不是的0, 1, 2, or 3数，作为要求的操作:

0	不允许恒定扫描模式
1	允许恒定扫描模式.
2	只设一个新的定时器值
3	读取恒定扫描模式和定时器值（看上面的注释）.

2. 时间值大于2550ms（2.55秒）.

3. 使能了一个没有编程的定时器为恒定扫描时间，或使用0旧值给定时器了。

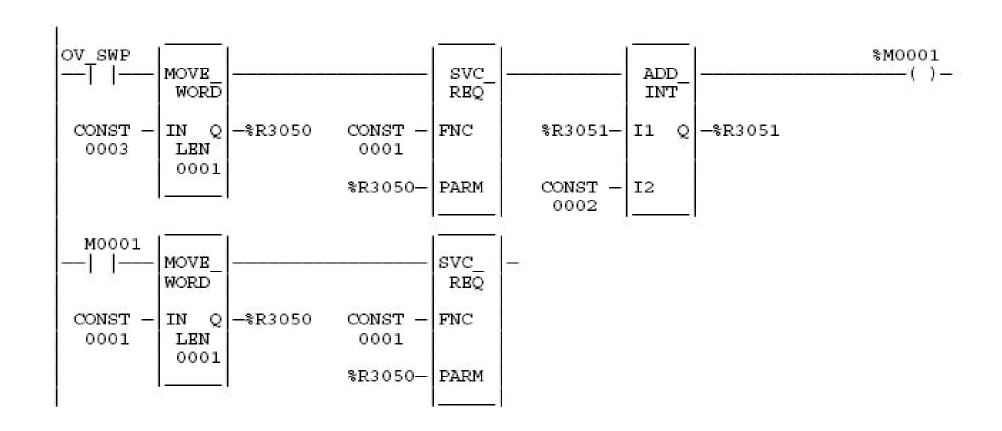
在函数执行后，函数反馈定时器的状态和值到相同的参数块参考地址处:

0 = 不允许 1 = 允许	地址
当前定时器值	地址 + 1

如果字地址+ 1 包含十六进制数FFFF, 没有定时器值被编译。

示例

本例子说明一个程序块中的逻辑。当OV_SWP 接点接通，读取恒定扫描定时器，定时器以2ms步长增加，新的定时器值反馈给PLC。参数块位于%R3050存储器内。因为MOVE和ADD功能需要3个水平接点位置，所有示例逻辑使用内部离散线圈%M0001作为临时位置，保持第一行的成功结果 。在任何扫描内OV_SWP 不设，%M0001关断。



SVCREQ #2: 读取窗口值

使用SVCREQ函数#2，对编程器通讯窗口和系统通讯窗口，取得当前窗口模式的时间值 。

注意

本手册讨论的CPU, 服务请求2 只被90-30CPU，8.0版以后的支持。

每个窗口有3个模式:

模式名称	值	说明
限制模式	0	窗口的执行时间受限制于各自的默认值或SVCREQ功能#3对编程器通讯窗口或SVCREQ功能#4对系统通讯窗口定义的值。当没有更多的任务需要完成时，窗口将终止。
恒定模式	1	每个窗口将在运行到结束模式下运行，PLC将在两个窗口切换，其时间是两个窗口各自时间的和。如果一个窗口是恒定模式，剩下的两个窗口自动设为恒定模式。如果PLC运行在恒定窗口模式，一个实际窗口的执行时间没有用相应的SVCREQ功能定义，窗口的默认时间做为恒定窗口时间计算。
运行到完成模式	2	不管窗口时间与一个实际窗口有关。是否故障或用一个服务请求功能定义，窗口将运行，直到窗口中的所有任务完成。

当时间值是0时，窗口不使能。参数块有3字长:

	高字节	低字节	
编程器窗口	模式	值（ms）	地址
系统通讯窗口	模式	值（ms）	地址+1
保留*	*看注释	*看注释	地址+2

***** 注释. 系统保留地址+ 2字使用。所有的0将反馈到这里。

所有参数是输出参数。没必要在参数块中输入值，以编译这个功能。所有敞口的输出值是毫秒。

示例

下面例子中，当使能输出%Q0102接通时，PLC操作系统将3个窗口的当前时间值放进%R0100开始的参数块中。其它展示读取窗口数值功能的例子包括在后面3个SYSREQ功能说明中。

```
|          _____
|%Q0102         |     |
|──| |───| SVC_|
|               | REQ |
|               |     |
|      CONST ─|FNC  |
|       0002   |     |
|              |     |
|     %R0100─|PARM |
|             |_____|
|
```

SVCREQ #3: 改变编程器通讯窗口模式和定时器值

使用 SVCREQ函数#3，改变编程器通讯窗口和定时器值。将在CPU调用功能的扫描周期内改变。

注意

本手册讨论的CPU, 服务请求3 只被90-30CPU，8.0版以后的支持。

SVCREQ函数#3将流通到右面输出，除非没有
选择0，1，2模式。参数块有1字长。.

为了不允许参数窗口，用如下的参数块输入SVCREQ函数#3:

高字节	低字节	
0	0	地址

为了允许参数窗口，用如下的参数块输入SVCREQ函数#3:

高字节	低字节	
模式	从1 到 255 ms的值	地址

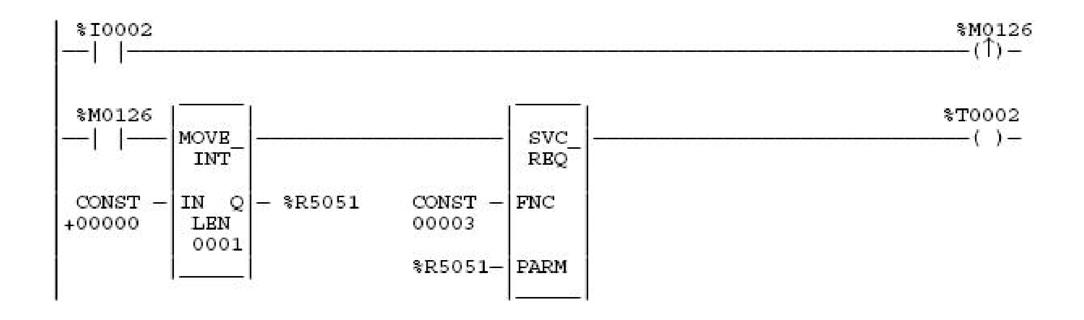
示例

下面的例子中，当 %M0125变为ON,编程器通讯窗口被允许，分配25ms值。参数块位于%R5051内存内。

```
 %I0001                                                              %M0125
--| |-----------------------------------------------------------------(↑)-

 %M0125   _______                          _______                   %T0002
--| |----|MOVE_  |------------------------|  SVC_ |--------------------( )-
         | INT   |                        |  REQ  |
 CONST  -|IN    Q|- %R5051      CONST    -|FNC    |
 +00025  | LEN   |              00003     |       |
         | 0001  |                        |       |
         |_______|              %R5051---|PARM   |
                                          |_______|
```

为了不允许编程器通讯窗口，使用服务请求3分配0值。在这个例子中，当当 %M0126变为ON,编程器通讯窗口被允许，分配0ms值。参数块位于%R5051内存内。



SVCREQ #4: 改变系统通讯窗口模式和定时器值

使用 SVCREQ函数#4改变系统通讯窗口模式和定时器值。将在CPU调用功能的扫描周期内改变。

注意

本手册讨论的CPU, 服务请求4只被90-30CPU，8.0版以后的支持。

SVCREQ函数#4将流通到右面输出，除非没有
选择0，1，2模式。参数块有1字长。

为了不允许参数窗口，用如下的参数块输入SVCREQ函数#4:

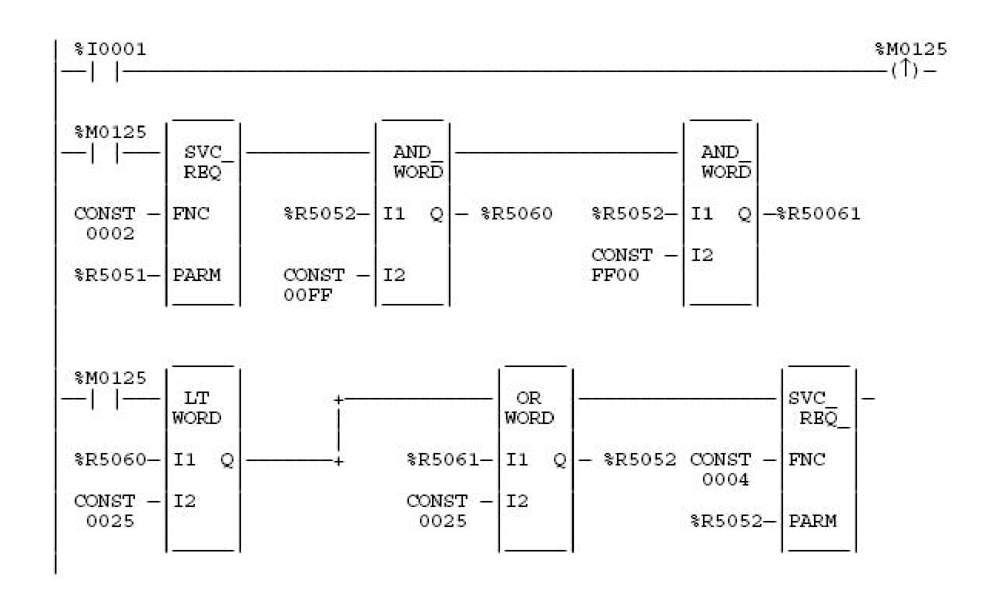
高字节	低字节	
0	0	地址

为了允许参数窗口，用如下的参数块输入SVCREQ函数#4:

高字节	低字节	
模式	从1 到 255 ms的值	地址

示例

如下例子中，当使能输出%M0125变为ON时，系统通讯窗口的模式和定时器值被读取。如果定时器值大于等于25ms,则值不变。如果比25ms小，则值变为25ms.另一种情况，当行执行完，窗口的使能。 所有3个窗口的参数块位于%R5051。由于系统通讯窗口的模式和定时器在参数块（读取窗口值功能，功能2#取得）的第2个值中，所以系统通讯窗口的现存窗口的时间位于%R5052的低字节。



SVCREQ #6: 改变/读取字数给检验和

使用SVCREQ函数，函数号为6。为了:

- 读 取 现 在 字 数 .
- 设 定 新 字 数 .

除非输入一些0和1以外的值作为请求的操作（看下面），将成功执行。

对于检验和任务功能，参数块有2字长。

为了读取当前字数:

用如下的参数块输入SVCREQ函数6:

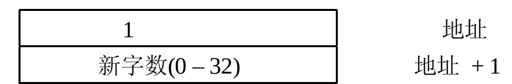
参数块	
0	地址
忽略	地址 + 1

在函数执行后，函数反馈当前检验和到参数块的第2个字中。对读取函数，没有指定范围；反馈值是现在被检验的字的数量。

参数块	
0	地址
现在字数	地址 + 1

为了设定新字数:

用如下的参数块输入SVCREQ函数6:



参数块	
1	地址
新字数(0 – 32)	地址 + 1

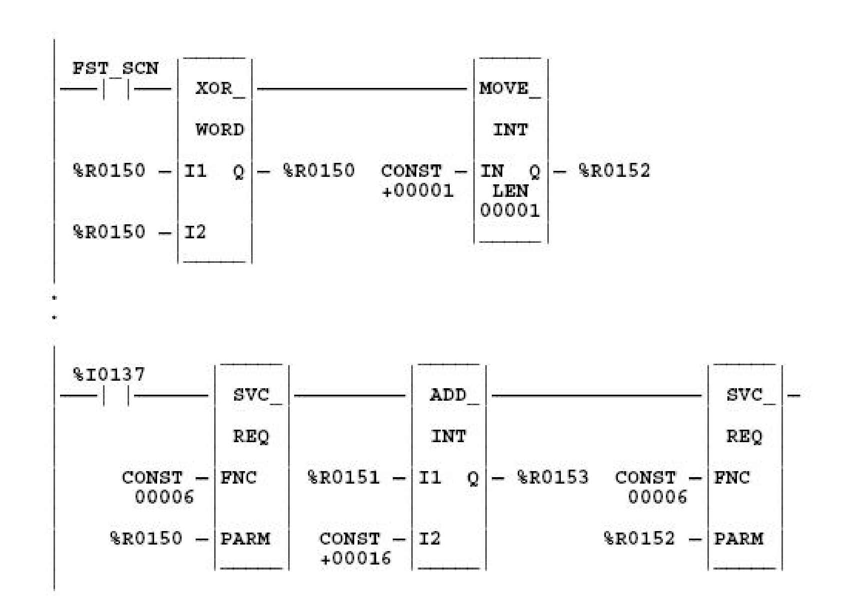
输入1，引起PLC调节校验字数与参数块内第2字内的值相同。对任何90-30系列 CPU,第2个字的值可以是 0到32. 如果值超过这个范围，将发生错误。对90-20系列CPU211，此值只能是0到4。

注意

这个服务请求不适用于Micro PLC.

示例

如下的例子中，当使能接点 FST_SCN为ON时，任务检验和功能的参数块就建立了。后面的程序，当输入%I0137为ON，被检验的字数从PLC操作系统读取。这个数字以16的间隔递增，ADD_INT功能的结果放进“保持新置位数”参数中。第2个服务请求块要求PLC设定新字数。



示例参数块位于地址 %R0150.它们有如下的内容:

0 =读取当前值	%R0150
保存当前值	%R0151

1 =设定当前值	%R0152
保存当前值	%R0153

SVCREQ #7: 改变/读取日历时钟

使用 SVCREQ函数，函数号为7，读取和设定PLC的日历时钟。

注意

本功能只适用于331或更高级的 90-30 CPU和28点的 90系列 Micro PLC CPUs(就是, IC693UDR005, IC693UAA007, 和 IC693UDR010)和23点的90系列Micro PLC CPU(IC693UAL006).

除了以下情况，将成功执行:

1. 一些不是0或1的数输入作为要求的操作（看下图）。
2. 指定了一个不可用的数据方式。
3. 提供的数据不是希望的方式。
4. 输入一个不可用的数据， 如02/29/01, 其不正确，因为2001年不是闰年。

对于日期/时间功能，参数块的长度取决于数据方式。BCD方式要求6个字；ASCII码要求12个字。

参数块	地址
0 = 读取时间和日期 1 = 设时间和日期	地址
1 = BCD 方式 3 = ASCII码方式	地址 + 1
数据	地址 + 2 到结束

字1内,指定功能是否读取或改变值.

0 = 读取
1 = 改变

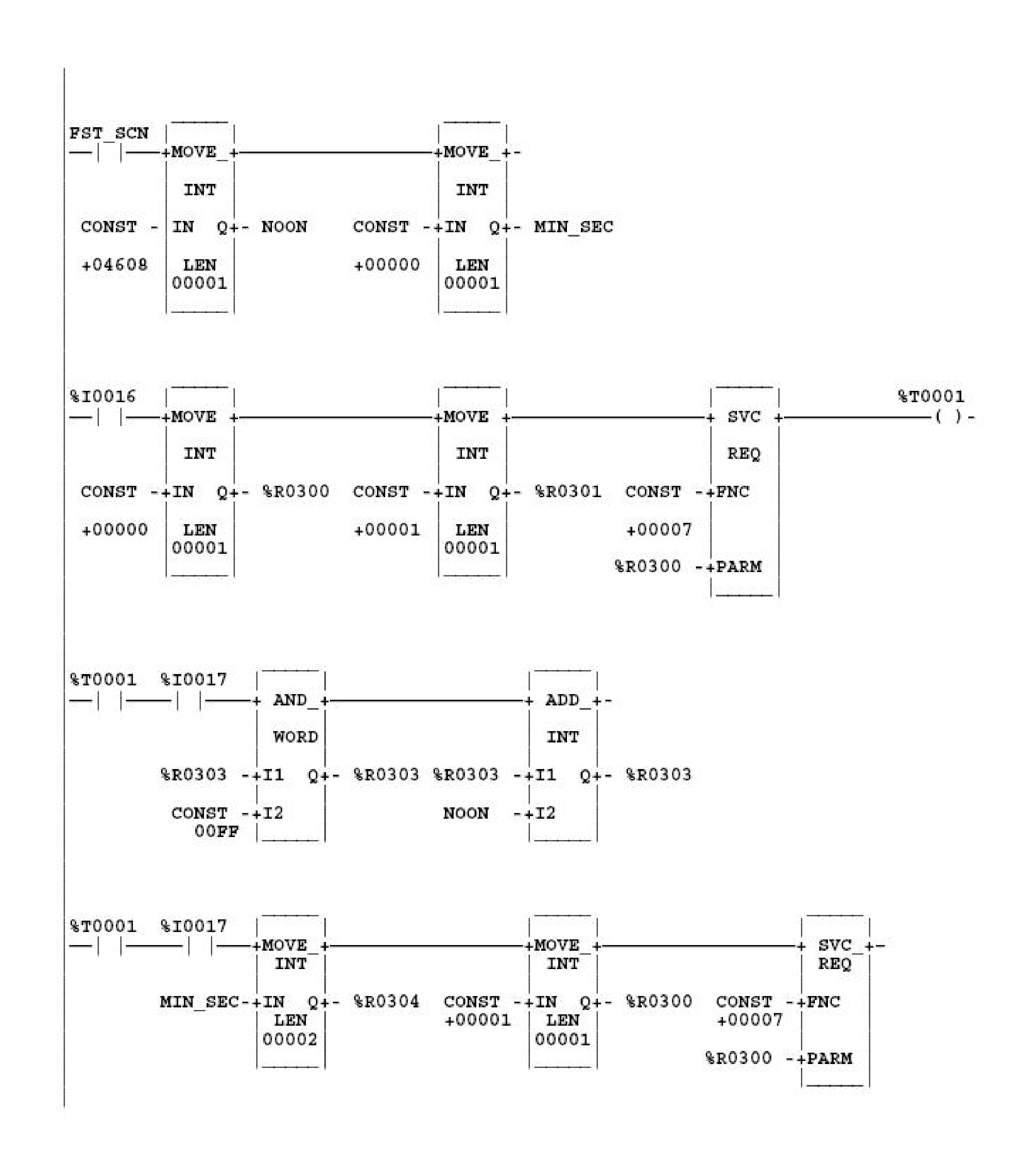
字2内,指定数据方式:

1 = BCD
3 = ASCII码，插入空格和冒号。

字3到参数块尾包含读取功能反馈回来的输出数据，或由改变功能提供新数据。在这两种情况，数据字的方式相同。当读取日期和时间时，参数块的字（地址+2）到（地址+8）忽略。

示例

下面例子中，当被先前的逻辑调用时，日历时钟的参数块被创建，首先要求当前日期和时间，然后以BCD方式设定时钟为下午12点。参数块位于全局数据%R0300. NOON列已经在程序的其他地方创建，包含值12，0和0。（NOON列必须包含%R0300的数据）。对于参数块，BCD方式要求6个连续的内存。



参数块内容

对于不同数据方式的参数块内容如下页所示。对所有数据方式:

- 小时按24小时方式存储。.
- 星期几是一个数字值:

值	星期几
1	星期天
2	星期一
3	星期二
4	星期三
5	星期四
6	星期五
7	星期六

用BCD方式改变/读取时期和时间:

BCD方式中，每个时间和日期项占一个字节。这种方式要求6个字。第6个字的最后字节不使用。当设定日期和时间，这个字节被忽略；当读取日期和时间，反馈回空字母（00）。

输出参数块例子：BCD方式读日期和时间
(Sun., July 3, 1988, at 2:45:30 p.m.)

高字节	低字节		输出例子(高)	输出例子(低)
1 = 改变 或 0 = 读取		地址	0	
1		地址 + 1	1	
月	年	地址+ 2	07	88
小时	号	地址+ 3	14	03
秒	分	地址+ 4	30	45
(空)	星期几	地址+ 5	00	01

使用ASCII码和空格冒号方式改变/读取日期和时间

ASCII码方式中，每个时间和日期项的数值是一个ASCII方式字节。另外，空格和冒号插入到日期中，允许其传输而不改变打印和显示设备。这种方式要求12个字。

输出参数块例子:
ASCII码方式读取日期和时间
(Mon, Oct. 2, 1989 at 23:13:00)

高字节	低字节		例子	
1 = 改变 或 0 = 读取		地址	0	
3		地址+ 1	3	
年	年	地址+ 2	39	38
月	(空)	地址+ 3	31	20
(空)	月	地址+ 4	20	30
号	号	地址+ 5	32	30
小时	(空)	地址+ 6	32	20
:	小时	地址+ 7	3A	33
分	分	地址+ 8	33	31
秒	:	地址+ 9	30	3A
(空)	秒	地址+ 10	20	30
号	号	地址+ 11	32	30

SVCREQ #8: 复位看门狗定时器

字扫描时，使用SVCREQ函数#8 ，复位看门狗定时器。

注意

本手册讨论的CPU, 服务请求8只被90-30CPU，8.0版以后的支持。

当看门狗定时器到期了，PLC关断，不给出警告。本功能允许定时器在一个时间消耗任务中继续运行（例如，当等待通讯反映时）。

警告

确定**重**新**启动看门狗**定时器不会影响控制**过**程。

这个函数与参数块无关；然而，编程软件要求一个空间给PARM。输入任何适当的参考地址；不使用它。

示例

下面的例子中，当 %Q0127 为ON时，看门狗定时器复位。

```
|                  _____
| %Q0127          |     |
|——| |———-————————| SVC_|—
|                 | REQ |
|                 |     |
|         CONST —|FNC  |
|          0008  |     |
|                |     |
|        %AI001—|PARM |
|                |_____|
|
```

SVCREQ #9: 从扫描的开始读取扫描时间

使用SVCREQ函数#9读取时间（ms），从扫描的开始起。数据是16位字方式。

注意

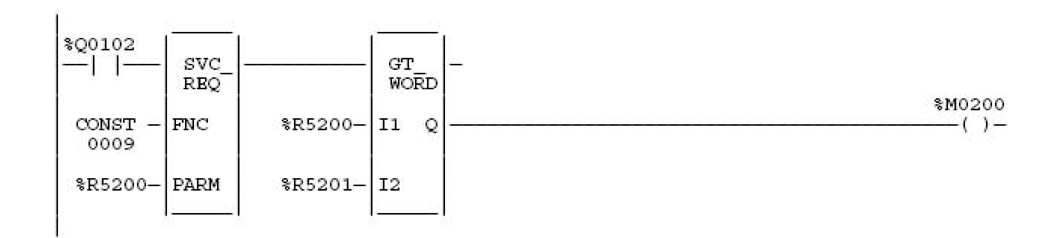
本手册讨论的CPU, 服务请求9只被90-30CPU，8.0版以后的支持。

参数块只是一个输出参数块，长度为一个字。

从扫描开始的时间	地址

示例

下面的例子中，从扫描开始的流逝时间总是存到%R5200.如果其比%R5201的值大，内部线圈%M0200变为ON 。



SVCREQ #10: 读取文件夹名称

使用SVCREQ 函数#10 读取当前运行文件夹的名称.

注意

本手册讨论的CPU, 服务请求10只被90-30CPU，8.0版以后的支持。

输出参数块有4个字长。反馈回8个ASCII字母；最后一个为空（00h） 如果程序名称比7个字母长，末尾为空字符。

低字节	高字节	
字母1	字母2	地址
字母3	字 母 4	地址 + 1
字母5	字 母 6	地址 + 2
字母7	00	地址+ 3

示例

下面的例子中，当使能接点 %I0301变到ON时，数值10装进寄存器 %R0099中，其是函数码，读取文件夹名称函数。随后的一行中，当 %I0102为ON,服务请求读取文件夹名称，存到 %R0100 (在 PARM指定) 开始的内存的4个字块中。

```
| %I0001                                                   %I0301
|--| |-----------------------------------------------------(↑)-
|
| %I0301  _______
|--| |---|MOVE_  |-
|         | WORD |
|         |      |
| CONST -|IN   Q|- %R0099
|  0010   | LEN  |
|         | 0001 |
|         |______|
|
|%I0102   _______
|--| |---| SVC_  |-
|         | REQ  |
|         |      |
| %R0099-|FNC   |
|         |      |
| %R0100-|PARM  |
|         |______|
|
```

SVCREQ #11: 读取 PLC ID

使用SVCREQ函数#11读取正运行程序的90系列PLC的名称。

注意

本手册讨论的CPU, 服务请求11只被90-30CPU，8.0版以后的支持。

输出参数块有4个字长。它反馈回8个ASCII字符；最后一个为空（00h） 如果PLC ID比7个字母长，末尾为空字符。

低字节	高字节	
字母1	字母2	地址
字母3	字母4	地址+ 1
字母5	字母6	地址+ 2
字母7	00	地址+ 3

示例

下面的例子中，当使能接点%I0001为OFF,数值11存入到寄存器%R0099 ，其是读取PLC ID函数的函数号。在随后的行中，当%Q0102 为ON, 服务请求读取PLC ID，将其存到从%R0100 (在PARM指定)开始的存储器的4个字块中。

```
|%I0001                                                          %M0301
|—| |————————————————————————————————————————————————————————————(↓)—
|
|          _____
|%M0301   |     |
|—| |——   |MOVE_|
|         | WORD|
|         |     |
|CONST  — |IN  Q|— %R0099
| 0011    | LEN |
|         | 0001|
|         |_____|
|
|          _____
|%Q0102   |     |
|—| |——   | SVC_|—
|         | REQ |
|         |     |
|%R0099— |FNC  |
|         |     |
|%R0100— |PARM |
|         |_____|
|
```

SVCREQ #12: 读取PLC运行状态

使用 SVCREQ 函数 #12读取PLC CPU的现在运行状态。

注意

本手册讨论的CPU, 服务请求12只被90-30CPU，8.0版以后的支持。

参数块只是一个输出参数块，有1个字长。执行这个服务请求，只产生2个可用的结果。

1 = 运行/不允许 2 =运行/允许	地址

示例

下面的例子中，当%I0102为ON时, 服务请求读取PLC运行状态，将结果放在寄存器%R402中。如果PLC处于运行/不允许模式，%R402将包含数值1。如果PLC处于运行/允许模式，%R402将包含数值2。

```
|            _____          _____
|%I0102     |      |
|--| |------| SVC_|-
|           | REQ  |         __________
|           |      |
|    CONST -|FNC   |
|      0012 |      |
|           |      |
|     %R402-|PARM  |
|           |_____|
|
```

SVCREQ #13: 中断 (停止) PLC

为了在下个扫描周期停止PLC，使用SVCREQ 函数#13。在下个PLC扫描周期的开始，所有输出将处于其指定的默认状态。一个故障信息将放入PLC故障表中，注意执行“中断PLC”函数块。I/O扫描将如配置继续扫描。

本函数没有参数块。

示例

下面的例子中，当一个“I/O模板丢失”故障发生时，SVCREQ函数#13 运行. 由于不需要参数块，不使用PARM输入; 然而，编程软件要求一个PARM 空间。

这个例子使用一个JUMP ，到程序末尾，如果中断PLC函数成功执行，强迫中断程序。JUMP和LABEL是必须的，因为直到函数执行的扫描周期末尾才转到停止模式。一旦PLC从服务请求收到停止指定，它将执行一个以上的扫描，然后停止（看下面的注释）。

```
|
|LOS_MD                                                              %T0001
|
|—| |——————————————————————————————————————————————————————————————————(↑)—
|
|              _____
|%T0001       |     |
|—| |————————| SVC_|———————————————————————————————————————>> END_PRG
|             |     |
|             | REQ |
|             |     |
|    CONST —|FNC  |
|      0013 |     |
|             |     |
|    %R1001 —|PARM |
|             |_____|
|
.
.
.
|
| END_PRG:
|
|
| [     END OF PROGRAM LOGIC    ]
|
```

注释

为了确保%S0002 LST_SCN 接点正确动作，PLC将在SVCREQ函数#13执行扫描周期后执行一个附加扫描周期。

SVCREQ #14: 清楚故障表

为了清楚PLC故障表或I/O故障表，使用SVCREQ 函数#14 。SVCREQ输出为ON，除非输入了0或1以外的数值作为请求的运行（看下面）。

对这个函数，参数块有一个字长。只是一个输入参数。

0 = 清楚PLC故障表. 1 = 清楚I/O故障表.	地址

示例

下面的例子中，当接点 %I0346 和 %I0349都是ON时，PLC故障表被清楚。当接点 %I0347 和 %I0349都是ON时，I/O故障表被清楚。当接点%I0348 和 %I0349 都是ON时，所有故障都被清楚。

PLC故障的参数块位于%R0500, I/O故障表的参数块位于%R0550. 在程序的任何地方都可以创建着两个参数块（为了清楚它们各自的表，它们都必须为逻辑1）。

```
|                           _____
|%I0349  %I0346            |     |
|--| |--+--| |--+----------| SVC_|-
|       |       |          |     |
|       |       |          | REQ |
|       |%I0348 |          |     |
|       +--| |--+CONST   --|FNC  |
|                0014      |     |
|                          |     |
|               %R0500   --|PARM |
|                          |_____|
|
|                           _____
|%I0349  %I0347            |     |
|--| |--+--| |--+----------| SVC_|-
|       |       |          |     |
|       |       |          | REQ |
|       |%I0348 |          |     |
|       +--| |--+CONST   --|FNC  |
|                0014      |     |
|                          |     |
|               %R0550   --|PARM |
|                          |_____|
|
```

SVCREQ #15: 读取故障表中最近的记录

为了读取PLC故障表或I/O故障表中最近记录的故障，使用SVCREQ 函数#15 。SVCREQ输出为ON，除非输入了0或1以外的数值作为请求的运行（看下面），或故障表为空。 (关于故障表内的其它信心，参考第3章，“故障说明和纠正”)。

对于这个函数，参数块长为22个字。输入参数块有这个格式:

输入参数块	
0 = 读取PLC故障表 1 = 读取I/O故障表	地址

输出参数块的格式取决于是否函数从PLC故障表或I/O故障表读数据。

PLC 故障表输出格式 / **I/O故障表输出格式**

地址	PLC 低字节	PLC 高字节	I/O 低字节	I/O 高字节
地址	0		1	
地址 +1	长/短		长/短	
地址+ 2	多余		参考地址	
地址+ 3	PLC 故障地址			
地址+ 4			I/O 故障地址	
地址+ 5	故障组和动作			
地址+ 6	错码		故障组和动作	
地址+ 7			故障类	故障类型
地址 +8			故障描述	
地址+ 9				
地址+ 10				
地址+ 11				
地址+ 12	故障特殊数据		故 障 特 殊 数 据	
地址+ 13				
地址+ 14				
地址+ 15				
地址+ 16				
地址+ 17				
地址+ 18				
地址+ 19				
地址+ 20	时间标记		时间标记	
地址+ 21				

地址+1字的第1字节总，长/短指示器定义故障内的故障特殊数据数量。可以是：

```
PLC 故障表:  00 = -8 字节 (短)
             01 = 24字节（长）
 I/O 故障表:  02 = —5 字节 (短)
              03 = 21 字节(长)
```

示例 1

下面的例子中，当输入%I0251为ON和输入 %I0250 为ON时，最后进入PLC故障表的故障被读取到参数块中。当输入%I0251为OFF和输入%I0250为ON时，最后进入I/O故障表的故障被读取到参数块中。参数块位于 %R0600.

```
|
|%I0250  %I0251  _____
|——| |—————| |———|MOVE_|
|                |     |
|                | INT |
|                |     |
|        CONST —|IN  Q|— %R0600
|          0000 | LEN |
|                | 0001|
|                |_____|
|
|%I0250  %I0251  _____
|——| |—————|/|———|MOVE_|
|                |     |
|                | INT |
|                |     |
|         CONST —|IN  Q|— %R0600
|          0001 | LEN |
|                | 0001|
|                |_____|
|
|ALW_ON    _____
|——| |———| SVC_|—
|          |     |
|          | REQ |
|          |     |
| CONST —|FNC   |
|   0015 |      |
|          |     |
|%R0600 —|PARM |
|          |_____|
|
```

示例 2

在下面的例子中，当I/O模板发生任何故障，除了在0机架中第9槽和1机架中第9槽模板发生故障外，PLC中断。如果这两个模板发生故障，系统继续运行。第1扫描周期内创建了“表类型”参数。接点IO_PRES,当设定了，表示I/O故障表为空。在故障逻辑输入一个故障到列表中后，在下个扫描周期PLC CPU设定常开接点为ON。如果连续2个扫描周期输入故障，2个扫描周期都设定常开接点。

例子使用一个位于%R0600的参数块。在SVCREQ函数执行后，参数块的第4个字包含有故障的I/O模板所在机架和槽位置：

1		%R0600
长/短		%R0601
参考地址		%R0602
机架号	槽号	%R0603
I/O总线号	总线地址	%R0604
点地址		%R0605

故障数据

程序中， EQ_INT块将表格中的机架/槽地址与十六进制常数比较。当故障点的机架/槽与上面的说明一样时，内部线圈%M0007 变为ON。如果线圈%M0007为ON，其常闭接点为OFF，防止中断。相反，如果线圈 %M0007为OFF，因为故障点在不同的模板，其常闭接点变为ON，发生中断。

```
|
|FST_SCN  _____
|--| |---|MOVE_|-
|        |     |
|        | INT |
|        |     |
| CONST -|IN  Q|- %R0600
|   0001 | LEN |
|        | 0001|
|        |_____|
|
|         _____
| IO_PRES|     |                                                       %T0001
|--| |---| SVC_|-------------------------------------------------------( )-
|        |     |
|        | REQ |
|        |     |
| CONST -|FNC  |
|   0015 |     |
|        |     |
|%R0600 -|PARM |
|        |_____|
|
|         _____
|%T0001  |     |
|--| |---| EQ_ |-
|        |     |
|        | INT |
|        |     |                                                       %M0007
|%R0603 -|I1  Q|-------------------------------------------------------( )-
|        |     |
|        |     |
| CONST -|I2   |
|   0109 |_____|
|
|         _____
|%T0001  |     |
|--| |---| EQ_ |-
|        |     |
|        | INT |
|        |     |                                                       %M0007
|%R0603 -|I1  Q|-------------------------------------------------------( )-
|        |     |
|        |     |
| CONST -|I2   |
|   0265 |_____|
|
|                  _____
| IO_PRES  %M0007 |     |
|--| |--------|/|---| SVC_|-
|                 |     |
|                 | REQ |
|                 |     |
|         CONST -|FNC  |
|           0013 |     |
|                 |     |
|         %R0001 -|PARM |
|                 |_____|
|
```

SVCREQ #16: 读取流逝时钟

为了读取系统的流逝时钟，使用SVCREQ 函数，函数号为16。从PLC上电开始，这个时钟按秒来跟踪流逝的时间。定时器大约每100年运行一圈。

这个函数只有输出参数块。参数块长度为3个字。

参数块	地址
从上电开始的秒数 (低序)	地址
从上电开始的秒数 (高序)	地址+ 1
100 微秒记号	地址+ 2

前两个字是流逝的时间（秒）。最后的字是现在秒内100微秒记号的数量。

示例

下面的例子中，当内部线圈%M0233为ON，读取流逝时钟的值，线圈%M0234为ON 。当为OFF时，再读其值。计算两个值的差，结果存到寄存器%R0250内。

第1次读取的参数块位于%R0127内；第2次读取的参数位于%R0131。这个计算忽略了100微秒记号的数量和DINT类作为有符号数计算。从上电后大约50年，计算将进行纠正。

```
|
|%M0233  _____                                                   %M0234
|--| |--| SVC_|--------------------------------------------------------(S)-
|        |     |
|        | REQ |
|        |     |
| CONST -|FNC  |
|  00016 |     |
|        |     |
|%R0127 -|PARM |
|        |_____|
|
|%M0233   %M0234 _____              _____                         %M0234
|--|/|--------| |--| SVC_|---------------| SUB_|-------------------------(R)-
|                  |     |               |     |
|                  | REQ |               | DINT|
|                  |     |               |     |
|           CONST -|FNC  |      %R0131 -|I1  Q|- %R0250
|            00016 |     |               |     |
|                  |     |               |     |
|          %R0131 -|PARM |      %R0127 -|I2   |
|                  |_____|               |_____|
|
```

SVCREQ #18: 读取I/O 待机状态

为了读取待机CPU的状态，使用SVCREQ 函数#18。

注意

这个功能仅适用于331或更高级的CPU。

关于这个函数，参数块长1个字。只有输出参数块。



0 = 非待机状态
1 = 待机状态

地址

注意

SVCREQ #18 只报道%I 和 %Q参考地址的待机状态。

示例

下面的例子中，I/O待机的状态总是存到%R1003中。如果存在任何待机，输出%T0001为ON。

```
|
|%I0001    _____            _____
|         |     |          |     |
|--|/|----| SVC_|----------| EQ_ |-
|         |     |          |     |
|         | REQ |          | INT |
|         |     |          |     |                                   %T0001
| CONST  -|FNC  |   CONST -|I1  Q|-----------------------------------( )-
|  00018  |     |   00001  |     |
|         |     |          |     |
|%R1003  -|PARM |  %R1003 -|I2   |
|         |_____|          |_____|
|
```

SVCREQ #23: 读取主校验和

使用SVCREQ 函数#23读取用户程序和配置的主校验和。如果函数被使能了，SVCREQ 输出总为ON，信息（看下面）的输出块从SVCREQ函数的参数3（PARM）开始。

当激活**RUN MODE STORE**,直到存储结束，程序校验和才能使用。因此，在输出参数块的开始给出2个标记，以表示程序和配置检验和可以用了。

对于这个函数，输出参数块有12字长，如下格式：

主程序检验和可用(0 = 不可用, 1 =可用)	地址
主配置检验和可用(0 =不可用, 1 =可用)	地址+ 1
程序块的数量 (包括主程序块)	地址+ 2
用户程序字节数(DWORD 数据类型)	地址+ 3
程序附加检验和	地址+ 5
程序CRC检验和 (DWORD 数据类型)	地址+ 6
配置数据字节数	地址+ 8
配置附加校验和	地址+ 9
配置CRC 校验和(DWORD数据类型)	地址+ 10

示例

下面的例子中，当输入%I0251为ON，主校验和信息放入参数块中，输出线圈(%Q0001)为ON。参数块位于%R0050。

```
|
|%I0251   _____                                          %Q0001
|--| |---| SVC_|-------------------------------------------( )-
|        |     |
|        | REQ |
|        |     |
| CONST -|FNC  |
|  +0023 |     |
|        |     |
|%R0050 -|PARM |
|        |_____|
|
```

SVCREQ #24: 复位智能模板

使用SVCREQ函数#24复位子板或智能模板。SVCREQ输出为ON，除非输入了一个不可用的机架或槽号，如下所示。

关于这个函数，参数块长为1个字。其只是一个输入参数块。

模板槽号 (低字节) 模板机架号(高字节)	地址

注意: 机架0，槽1表示发送给子模板一个复位信号。

示例

下面例子中，当输入%I0346为ON和输入 %I0349为ON时，内%R0500的机架/槽指定的模板复位。

参数块包含服务请求复位的模板的机架和槽，位于%R0500 中。在程序的其它地方创建参数块。

```
|
|%I0349  %I0346        _____
|——| |——+——| |——+———————| SVC_|—
|       |       |       |     |
|       |       |       | REQ |
|       |%I0348 |       |     |
|       +——| |——+CONST —|FNC  |
|                0024   |     |
|                       |     |
|              %R0500 —|PARM  |
|                       |_____|
|
|                     _____
|
```

SVCREQ #26/30: 访问 I/O

如果保存配置已经完成，使用SVCREQ 函数#26 (或#30—它们一样； 你可以使用任意一个完成同样的事情)访问实际存在的模板，将它们与机架/槽配置相比较，产生的附加信息，丢失，和不匹配报警。依据那个故障，SVCREQ将在PLC和I/O故障表内发出故障。

本函数没有参数块，输出总为ON。

注意

SVCREQ的运行时间取决于存在多少故障。因此，如果故障模板越多，SVCREQ的运行时间越长。

例子

下面例子中，当输入%I0251为ON，访问实际的模板，与机架/槽配置比较。在SVCREQ完成后，输出%Q0001为ON。

```
|
|%I0251   _____
|         |     |                                               %Q0001
|--| |----| SVC_|-----------------------------------------------( )-
|         |     |
|         | REQ |
|         |     |
| CONST  -|FNC  |
|   +0026 |     |
|         |     |
|%R0050  -|PARM |
|         |_____|
|
```

注意

本服务请求不适用于Micro PLC.

SVCREQ #29: 读取流逝的电源切断时间

使用SVCREQ 函数#29读取上次掉电与最近上电间流逝时间的总数。SVCREQ输出总为ON,信息（看下面）的输出块从SVCREQ功能的参数3（PARM）给定的地址开始。

注意

这个功能仅适用于331或更高级的CPU。

本函数只有输出参数块。参数块长为3个字。

参数块	地址
电源切断流逝秒数(低序)	地址
电源切断流逝秒数(高序)	地址+ 1
100 微秒记号	地址+ 2

前两个字是电源切断流逝的时间（秒）。最后的字是以100微秒记号（总是0）保持的电源切断流逝的时间。无论什么时候，PLC不能完全计算电源切断流逝的时间，时间将设为0。当PLC再次上电时（CLR M/T加给HHP）是，其将发生。如果在电源切断前看门狗定时器时间溢出，其也将发生。

示例

下面例子中，当输入 %I0251为ON时，流逝的电源切断时间放入参数块中，输入线圈(%Q0001)变为ON。参数块位于%R0050。

```
|
|%I0251   _____                                                %Q0001
|--| |---| SVC_|-------------------------------------------------( )-
|        |     |
|        | REQ |
|        |     |
| CONST -|FNC  |
|  +0029 |     |
|        |     |
|%R0050 -|PARM |
|        |_____|
|
```

SVCREQ #45: 跳到下个输出 & 输入扫描

(悬挂I/O) 使用SVCREQ 函数#45跳到下个输出和输入扫描。在SVCREQ#45运行扫描周期中，输出参考地址的任何改变将不影响相应模块的物理输出。 在SVCREQ#45运行后的一个扫描周期内，模板上的物理输入的任何变化将不会影响相应的输入参考地址。

本函数没有参数块.

注意

DOIO函数块不受使用SVCREQ#45的影响。当与SVCREQ#45一起用在同一逻辑程序中，它将继续更新I/O。

示例

在下面的例子中，当“Idle”接点导通时，下面的输出和输入扫描被跳过。

IDLE

SVC_
REQ

CONST
00045

FNC

%R0001

PARAM

SVCREQ #46: 高速底板状态访问

本函数一种通讯方法，经过PLC底板与智能模板进行高速通讯（与它通讯方式相比） 。靠限制数据量和回答数量，达到更大的通讯速率。

使用SVCREQ 函数#46完成一个如下的高速底板访问函数:

- 读取指定智能模板的额外数据字。
- 写指定智能模板的额外数据字。
- 读取/写: 读取指定智能模板的额外数据字，在一个操作中，将0和15间的数据写到同样的模板。

注意

现在，唯一支持本服务请求的模板是DSM314 (数字伺服模板).

在同一逻辑扫描周期（任意数据写函数完成了），COMM_REQ 或DOIO函数块应该不能执行指定的模板。因为它们会引起写的数据丢失。.

在同一逻辑扫描周期，写到模板的2个函数（写或读/写）应该不能执行同一模板，因为它们会引起第1写入数据丢失。

本服务请求也称“SNAP.”

本服务请求有如下所示的可变长度。参数块的第1个字决定将使用哪个函数，如下格式:

1 = 读额外数据 2 = 写额外数据 3 = 读/写额外数据	地址(字1)

读额外状态数据(功能 #1)

读额外数据函数从每个模板读额外状态数据的一个字，这个模板在参数块中的列表指定，将状态数据值放进参数块中。参数块要求(N + 4)个字的内存，N是将要被写数据的模板数。

使用下面页的表，说明输出值。

表 12-5. 读额外数据函数的参数块

位置	内容	意思
地址	函数	1 =读额外状态数据
地址+ 1	错误代码	如果因为不存在任何模板，不相称，或不工作而函数失败，一个错误代码放在这里。更详细的说明，看12-75的 “错误代码”。
地址+ 2	错误机架&槽	发生错误的机架号& 槽号。
地址+ 3	第一机架&槽	第1个将被读的模板的机架和槽号(十六进制的RRSS内, RR 是机架号， SS是槽号)
地址+ 4	从第1模板读数据	从第1模板读的数据放在这里。
地址+ 5	第二机架 & 槽	第2个将被读的模板的机架和槽号(十六进制的RRSS内, RR 是机架号， SS是槽号)
地址+ 6	从第二模板读数据	从第2模板读的数据放在这里
地址+ (I * 2) + 1	第 I机架 & 槽	第I个将被读的模板的机架和槽号(十六进制的RRSS内, RR 是机架号， SS是槽号)
地址+ (I * 2) + 2	从第I模板读数据	从第I模板读的数据放在这里。
地址+ (N * 2) + 1	最后机架&槽	最后1个将被读的模板的机架和槽号(十六进制的RRSS内, RR 是机架号， SS是槽号)
地址+ (N * 2) + 2	从最后模板读数据	从最后模板读的数据放在这里。
地址+ (N * 2) + 3	列表结束指示	这个字内输入0，表示模板列表结束。

写数据(功能 #2)

写数据函数将0和15间的一个数据值从参数块写到参数块列表指定的1个以上的模板。参数块要求（N+4）字的内存，N是数据写入的模板数。

表 12-6. 写数据**函**数的参数块

位置	内容	意 思
地址	函数	2 = 写数据
地址+ 1	错误代码	如果因为不存在任何模板，不相称，或不工作而函数失败，一个错误代码放在这里。如果函数执行了，但没有任何模板正确收到写的数据，不会有错误代码产生。更详细的说明，看12-75的 “错误代码”。
地址+ 2	错误机架&槽	发生错误的机架&槽号
地址+ 3	第一机架&槽	数据发送的第1个模板的机架和槽号(十六进制的RRSS内, RR 是机架号， SS是槽号)
地址+ 4	写数据到第1模板	将这个数据值写给第1模板
地址+ 5	第2机架&槽	数据发送的第2个模板的机架和槽号(十六进制的RRSS内, RR 是机架号， SS是槽号)
地址+ 6	写数据到第2模板	将这个数据值写给第2模板
地址+ (I * 2) + 1	第I机架&槽	数据发送的第I个模板的机架和槽号(十六进制的RRSS内, RR 是机架号， SS是槽号)
地址+ (I * 2) + 2	写数据到第I模板	将这个数据值写给第I模板
地址+ (N * 2) + 1	最后架&槽	数据发送的最后模板的机架和槽号(十六进制的RRSS内, RR 是机架号， SS是槽号)
地址+ (N * 2) + 2	写数据到最后模板	将这个数据值写给最后模板
地址+ (N * 2) + 3	列表结束指示	这个字内输入0，表示模板列表结束。

读/写数据(功能 #3)

读/写函数从参数块指定模板读额外状态数据的一个字，于是将0和15间数据值从参数块写如模板。按参数块列表，重复读写每个模板。内存的参数块(N * 3) + 3字，N是数据将被交换的模板号。

表12-7. 读/写数据**函**数的参数块

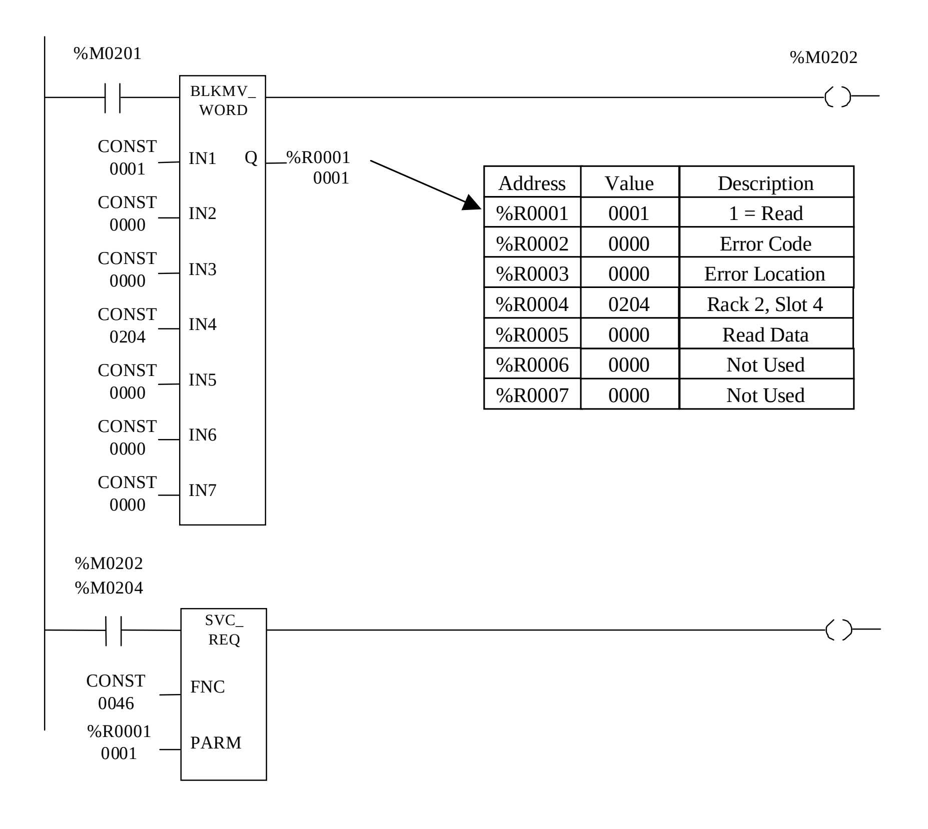
位置	**内容**	**意**思
地址	函数	3 =读/写
地址+ 1	错误代码	如果因为不存在任何模板，不相称，或不工作而函数失败，一个错误代码放在这里。如果函数执行了，但没有任何模板正确收到写的数据，不会有错误代码产生。更详细的说明，看12-75的 “错误代码”。
地址+ 2	错误机架&槽	发生错误的机架&槽号
地址+ 3	第一机架&槽	交换数据的第1个模板的机架和槽号(十六进制的RRSS内，RR 是机架号， SS是槽号)
地址+ 4	从第1模板读数据	从第1模板读的数据放在这里
地址+ 5	写数据到第1模板	将这个数据值写给第1模板
地址+ 6	第2机架&槽	交换数据的第2个模板的机架和槽号(十六进制的RRSS内，RR 是机架号， SS是槽号)
地址+ 7	从第1模板读数据	从第1模板读的数据放在这里
地址+ 8	写数据到第2模板	将这个数据值写给第2模板
地址+ ((I-1) * 3) + 3	第I机架&槽	交换数据的第I个模板的机架和槽号(十六进制的RRSS内，RR 是机架号， SS是槽号)
地址+ ((I-1) * 3) + 4	从第I 模板读数据	从第1模板读的数据放在这里
地址+ ((I-1) * 3) + 5	写数据到第I模板	将这个数据值写给第I模板
地址+ ((N-1) * 3) + 3	最后架&槽	交换数据的最后模板的机架和槽号(十六进制的RRSS内，RR 是机架号， SS是槽号)
地址+ ((N-1) * 3) + 4	从最后模板读数据	从最后模板读的数据放在这里
地址+ ((N-1) * 3) + 5	写数据到最后模板	将这个数据值写给最后模板
地址 + (N * 3) + 3	列表结束指示	这个字内输入0，表示模板列表结束

表 12-8. 错误代码

值	说明
1	成功 — 函数正常执行.
-1	指定槽中不存在模板.
-2	模板不相称 — 指定槽中的模板不是智能模板 或不支持本函数.
-3	模板不工作 — 指定槽中的模板不与CPU正常通讯
-4	读数据奇偶错误 —在读扩展或远程机架操作时, 发生奇偶错误。
-5	控制块中指定的不可用函数.

示例 1

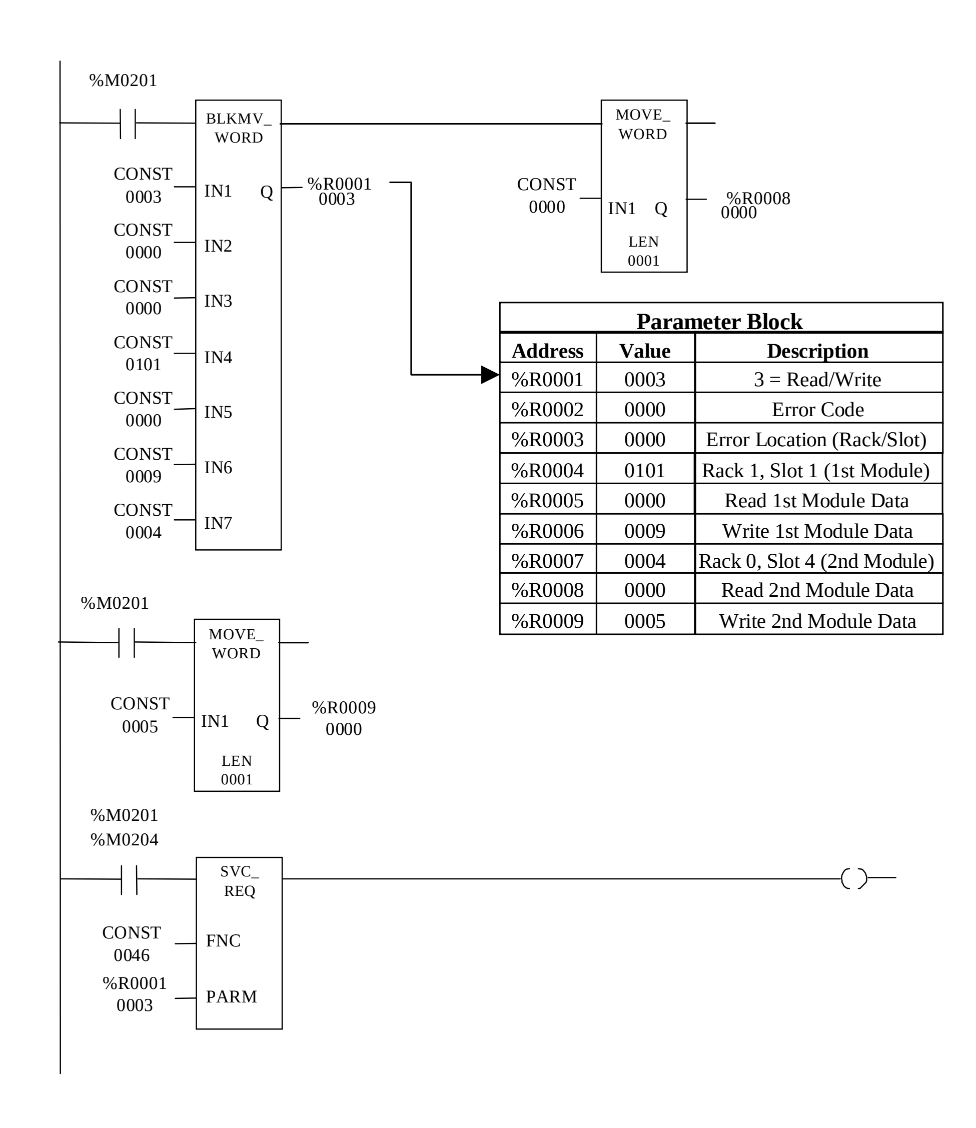
下面的例子展示了读取（%R0001中指定）一个单一模板，位于2机架，4槽（%R0004中指定）。如果函数成功地完成，读的数据存到%R0005中。如果发生错误，不管怎么样，错误代码将存入%R0003中。注意，由于对一个单一模板读函数，不使用地址+5和地址+6。因此，相应的内存， %R0006 和 %R0007, 填充为0（从BLKMV指令的IN6和IN7输入）。如果一个其它模板被读取， %R0006 和 %R0007 将用于其它模板。读函数的更多信息，看本章的表12-5。



Address	Value	Description
%R0001	0001	1 = Read
%R0002	0000	Error Code
%R0003	0000	Error Location
%R0004	0204	Rack 2, Slot 4
%R0005	0000	Read Data
%R0006	0000	Not Used
%R0007	0000	Not Used

示例 2

这个例子中，BLKMV 和两个MOVE 指令将要求数据写到参数块中，从%R0001 (SVCREQ PARM输入指定)开始。当使能时，SVCREQ 读取额外状态字数据，从机架0，槽4模板和机架1，槽1模板。将值0005 写到机架0，槽4模板，0009 写到机架1，槽1模板。(注意，模板没必要按槽号依次列入参数块。) 从机架0，槽4读的数据放入%R0008。从机架1，槽1读的数据放入%R0005.



Parameter Block		
Address	Value	Description
%R0001	0003	3 = Read/Write
%R0002	0000	Error Code
%R0003	0000	Error Location (Rack/Slot)
%R0004	0101	Rack 1, Slot 1 (1st Module)
%R0005	0000	Read 1st Module Data
%R0006	0009	Write 1st Module Data
%R0007	0004	Rack 0, Slot 4 (2nd Module)
%R0008	0000	Read 2nd Module Data
%R0009	0005	Write 2nd Module Data

SVCREQ #48: 致命故障自动复位后重新启动

SVCREQ 48兼容性

CPU – 10.00版 (或更高版本) 90-30系列 CPUs 331, 340, 341, 350, 36x, 和37x固件支持本服务请求。

Software – 只有VersaPro 1.1版 (或更高版本) PLC软件支持本服务请求。 Logicmaster不支持本特性。

警告

在应**用软件中，**致命故障**后**重**新**启动特**性不**应该**使用**（忽略致命故障**参数设为不**允许）**，在**故障情况**下**自动**PLC重**新**启动**，控制设**备**内可能产生不**安**全**情况**。**决**定**是否对他们**的设**备**使用这个**特**性，**是**系统设**计员**的**责任**。**紧随**这个警告的**故障**可能**导致人员伤害**或**死亡/或**损坏**设**备**。**

讲述

在发生致命故障后，致命故障后重新启动服务请求使PLC自动恢复正常操作。下面的致命故障，PLC将自动复位，恢复执行。故障将不被清楚，但将以非致命性处理。如果上电后存在致命故障，PLC 将仍然允许过度到运行模式。经在CPU硬件配置中忽略致命故障（或跳过致命故障）参数，使能这个特性。

SVCREQ 48 设定最大数量的重试和时间周期，在这期间发生重试。如果在这个时间内允许重试的数量超过了，CPU模式设定为STOP/FAULT。如果周期为0，当允许重试的数量超过了，CPU模式设定为STOP/FAULT。

如果操作员恢复电源，致命故障被忽略。现在故障值和时间周期初始化。致命故障的总数不变，但重试的总数递增。无论什么时候重试成功，保持设定直到所有致命故障被清楚，或为STOP/FAULT模式。系统位%S0021变为1。

Table 12-9. **致命故障**后**重**新**启动**的参数块

位置	内 容	意 思
字1	服务请求状态	看返回状态定义，下面. 用户程序必须初始化这个字为0。
字2	没限制的重试	0 = 不允许(字3设定重试的数量) 1 = 允许 (忽略字3和4)
字3	允许重试的数量	范围0到128 0 = 不允许自动重新启动 1 到 128 = 字4设定周期内允许的重试的最大数量。
字4	重试周期 (分钟)	范围0到5940分钟(99小时) 0 = 没有时间限制，字3中设定的重试的最大数量。自动重新启动将允许重试的数量。 1 到5940 = 如果指定的重试数量超过指定的周期，不允许自动重新启动。

Table 12-10. **致命故障**后**重**新**启动返回**状态说明

状态	描述	注释	电源流
-5	不可用的重试周期	0到5940为可用范围	No
-4	不可用的重试数量	0到128为可用范围	No
-3	不可用的无限制重试	必须是0或1	No
-2	配置不允许	在硬件配置中, 忽略致命故障(跳过致命故障) 必须设为允许	No
0	不动作	命令请求没变化	Yes
1	允许自动复位	有用命令允许致命故障后重新启动	Yes
2	不允许自动复位	有用命令允许致命故障后重新启动 允许保持忽略致命故障.	

SVCREQ 49 自动重启统计

服务请求49提供2个选项，记录发生致命故障和重新启动的总数。值的范围是0到65535. 如果超过它们的最大值，这些值不滚动计数。(服务请求48 用来定义允许重新启动的最大数和发生重试的时间限制。)

表 12-11. 自动重启统计的参数块

字 1	服务请求状态	看下面返回状态定义.
字 2	命令	用户程序必须初始化这个字为0。 0 = 返回致命故障的总数和已经发生的重试的数量。 1 = 初始化致命故障的总数和重试的数量为0。.
字3	返回的值= 已经发生的致命故障的总数。.	用户程序应该初始化为0。
字4	返回的值= 自动复位重试的总数。	用户程序应该初始化为0。

Table 12-12. 自动重启统**计**的**返回**状态定**义**

状态	描述	注释	电源流
-2	不允许配置	在硬件配制中，忽略致命故障(跳过致命故障) 选项必须设为允许。	No
-1	不可用的命令	命令必须是0或1。	No
1	正常状态	可用命令	Yes

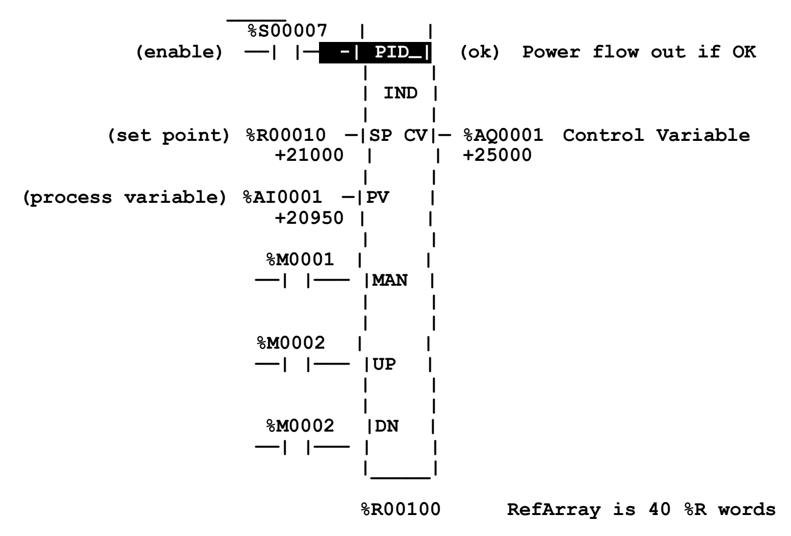
CPU兼容SVCREQ 49

10.00版 (或更高版本) 90-30系列 CPUs 331, 340, 341, 350, 36x, 和37x固件支持本服务请求。

PID

比例加上积分加上微分(PID)控制函数是最知名的普通运算法则，用于闭环控制。90系列函数块将一个输出变量（PV）反馈与一个设定变量（SV）进行比较，根据这个差值，更新控制变量（CV）输出。

块使用PID环增益和其它参数，这些参数存在一个4016位字的列中（12-82页中有所论述），在希望的时间间隔解决PID运算。所有参数是16位整形字，同16位类似过程变量组合而成。允许使用%AI 内存作为输入过程变量，%AQ作为输出控制变量。下面的例子包括典型输入。



(参考列地址)

由于按16整形数度量，许多参数必须在8PV数或单位或CV数或单位中定义。例如，SP输入必须如PV一样度量，因为PID块按这两个输入的差额来计算误差。PV和CV数可以是–32000 或 0到 32000,匹配的类似度量或从0到10000，显示的数值是0.00%到 100.00%。PV和CV计数并不是必须一样度量，在这种情况下，PID增益中将有一个度量系数。

注意

PID将不会比10ms更快执行。如果你设定每个扫描执行，而扫描时间不足10ms，将改变你的结果。在这种情况，PID函数将不运行，直到累积起了足够的扫描时间，达到10ms。例如，如果扫描时间是9ms，PID函数将使下一次扫描流逝时间达到18ms才执行一次。

参数

参数	说明
enable	当经过一个接点允许，PID函数执行。.
SP	SP是控制环或过程设定点。用PV计数设定，PID调节输出CV，于是PV达到SP（0误差）。
PV	过程变量经正在控制的过程输入，经常是 %AI输入.
MAN	当激活为1（经过1个接点），PID块为手动模式，如果这个参数没有被激活（0），PID块为自动模式。
UP	如果沿着MAN激活，其调节CV，每次成倍增加*。
DN	如果沿着MAN激活，其调节CV，每次成倍减少*。
RefArray Address	地址是PID控制块信息（使用和内部参数）的威势，使用40 %R 字，其不能与其它共用。
ok	ok输出被激活，当函数完成，且没有错误。 如果存在错误，其为OFF。
CV	CV 是控制变量输出给过程，经常是 %AQ 模拟输出.

*增加(UP 参数) 或减少 (DN 参数)，按PID函数的每次调用。

可用的内存类型

参数	不定	%I	%Q	%M	%T	%S	%G	%R	%AI	%AQ	常数	无
enable	•											
SP		•	•	•	•		•	•	•	•	•	
PV		•	•	•	•		•	•	•	•		
MAN	•											
UP	•											
DN	•											
address								•				
ok	•											•
CV		•	•	•	•		•	•	•	•		

- 可用参考或地址，能流能流过函数。

PID 参数块

除了2个输入字和3个手动控制接点，PID块使用RefArray中参数13。在调用这个块前，这些参数必须设定。其它参数被PLC使用，不用配置。在下表中出的%Ref与PID块底下的RefArray地址一样。加号后面的数值是列中的偏移量。例如，如果RefArray在%R100开始，%R113将包含手动命令（设定控制变量和控制模式的积分器）。

表 12-13. PID 参数概述

寄存器	参 数	低位单位	值范围
%Ref+0000	循环数	整 型	0到255 (仅显示)
%Ref+0001	运算法则	N/A; 由PLC 设定和维持	不可配
%Ref+0002	采样周期	10 ms	0 (每个扫描) 到 65535 (10.9 Min). 对于90-30PLC，至少使用10 (看12-80页的注释).
%Ref+0003	死区 +	PV计数	0 到 32000 (无负)
%Ref+0004	死区 —	PV 计数	–32000 到0 (无正)
%Ref+0005	比例增益 –Kp	0.01 CV%/PV%	0 到 327.67 %/%
%Ref+0006	微分增益–Kd	0.01 秒	0 到 327.67 sec
%Ref+0007	积分速率–Ki	重复/1000 Sec	0 到32.767 次数/sec
%Ref+0008	CV 偏差/输出偏移	CV计数	–32000 到 32000 (加到积分器输出)
%Ref+0009	上限	CV计数	–32000到32000 (>%Ref+10)输出限制
%Ref+0010	下限	CV计数	–32000到32000 (<%Ref+09)输出限制
%Ref+0011	最小回转时间	秒/ 满程	0 (无) 到32000 秒，移动32000 CV
%Ref+0012	定义字	使用低5位	位 0 到 2 用于误差 +/–, 输出极性，微分。
%Ref+0013	手动命令	CV计数	自动时跟随CV或手动时设定CVl
%Ref+0014	控制字	由PLC控制，除非位1被设定。	由PLC控制, 除非设定其它: 低位设定跳过1 (看12-85页内“PID 参数详细说明” 表中的说明
%Ref+0015	内部SP	N/A; 由PLC设定和维持。	不可配
%Ref+0016	内部CV	N/A; 由PLC设定和维持。	不可配
%Ref+0017	内部PV	N/A; 由PLC设定和维持。	不可配
%Ref+0018	输出	N/A; 由PLC设定和维持。	不可配e

表 12-13. PID参数概述- 续

寄存器	参 数	低位单位	值范围
%Ref+0019	微分项存储	N/A; 由PLC设定和维持。	不可配e
%Ref+0020 和 %Ref+0021	微分项存储	N/A; 由PLC设定和维持。	不可配
%Ref+0022	回转项存储	N/A; 由PLC设定和维持。	不可配
%Ref+0023	时钟	N/A;	
%Ref+0024		由PLC设定和	不可配
%Ref+0025	(执行持续时间)	维持。	
%Ref+0026	Y 剩余存储	N/A; 由PLC设定和维持。	不可配
%Ref+0027	SP，PV的低范围	PV 计数	–32000到 32000 (>%Ref+28) 用于显示
%Ref+0028	SP，PV的高范围	PV计数	–32000 到 32000 (<%Ref+27) 用于显示
%Ref+0029 +• %Ref+0034	为内部使用保留	N/A	不可配
%Ref+0035 • %Ref+0039	为外部使用保留	N/A	不可配

在90-30PLC上，RefArray列必须有%R寄存器组成。注意，每个PID块调用必须使用不同的40字列，即使所有13用户程序相同，但列中的其它字用于内部PID数据存储。确保列不会超过内存的末尾。

为了配置运行参数，选择PID函数，按F10键，进入用户参数屏；于是使用方向键选择区域，输入希望的数值。你可以使用0作为大部分默认值，除了CV上限，其必须比CV下限大，以保证PID块运行。注意，如果在用户参数中有错误，PID块不通过，所有当修改数据时，用一个临时线圈监视。

一旦选定适合的PID数值，它们应该在BLKMOV中定义为常数。如果需要，可以使用调用默认PID用户参数。

PID指令的运行

平常自动化运行是调用PID块，每个扫描周期，并且有能流使能，手动输入接点没有能流。块将现在PLC流逝时间与存在内部RefArray中最近PID解答时间进行比较。如果时间差比RefArray内第3个字(%Ref+2) 定义的采用周期大，PID运算法则使用这个时间差解答，最近的解答时间和控制变量输出更新。在自动模式中，输出控制变量放在手动命令参数%Ref+13中。

如果允许和手动输入接点都有能流，PID块放入手动模式，输出控制变量由手动命令参数%Ref+13设定。如果UP或DN输入有能流，手动命令字是递增或递减，每个PID解答按一个CV计数。为了输出控制变量的更快地手动改变，直接加或减任何CV计数到手动命令字是可能的。

PID块使用CV上限和下限参数限制CV输出。如果定义了一个正最小回转时间，其用来限制CV输出的斜率。如果超过CV幅值或斜率，调整保存在积分器的值，使其等与限制值。这个反对自设定终止特征（12-87页定义）意思是，即使误差很长时间内努力驱动CV超过（低于）限制值，一旦误差改变极性，CV输出将马上离开限制值。

本操作，同自动模式中的手动命令跟踪CV和手动模式中设定的CV，在自动和手动模式间提供一个平滑变换。自动模式下，CV上限和下限和最小回转时间仍然作用于CV输入，保存在积分器值的内部数值更新。这意思是说，如果你在手动模式，想离开手动命令，CV输出将不比最小回转时间斜率更快地改变，将不会比超过CV上限或低于CV下限。

注意

每个扫描周期中，调用一个特殊的PID函数不应该多于1次。

下表提供更详细内容，关于12-3表中简单讨论的参数。每个参数名后括号内的数值是RefArray 的偏移量。

表12-14. PID 参数详述

数据项	说 明
循环数 (00)	这是一个可选择的参数，用来确定一个PID块。其是一个无符号整数，在PLC中提供一个公共的证明，用循环数（由操作接口设备定义）。当从Logicmaster 90-30/20/Micro软件监视程序时，循环数在块地址下显示。
运算法则(01)	由PLC设定的一个无符号整数确定函数块使用什么运算法则。ISA运算法则定义为运算法则1，独立的运算法则定义为2。
采样周期 (02)	最短时间，按10ms递增，PID运算的解答间。例如，使用10表示100ms采样周期。如果设定为0，每次块被调用（看下面的PID块进度部分），运算都执行。 只有当前PLC流逝时间比最近PID解答时间加这个扫描周期都迟，PID运算才进行。记住，90-30将不使用比10ms(看12-82页的注释)还短的计算时间;因此扫描将跳过更小的扫描时间。这个函数用100ms补偿由最近执行流逝的时间。如果这个值设为0，函数每次被使能，其就执行；然而，其被限制到如上面注释所讲的最小10ms。
死区 (+/—) (03/04)	INT 值定义PV计数内上(+)和下 (–) 死区限制。如果没有死区要求，这些值必须是0。如果PID误差（SP-PV）或（PV-SP）比（-）值大，比（+）值小，PID以0误差来计算。如果没有0， (+)值必须比0大，(–)值必须比0小，或PID块将不运行。你应该使这些为0，直到PID循环增益被设定或调节。在这后，你可以加入死区，以避免由于误差小的变动而引起小的CV输出，可以减少机械磨损。
比例增益–Kp (05)	这个INT 数, 称为控制器增益, Kc, ISA翻译中, 用来确定CV值的变化，误差项中，对于100PV计数变化。以0.00 %/%形式显示，有2位小数。例如，Kp输入450，将显示为4.5,结果是Kp*误差/100或450*误差/100 作为PID输出的精度。当调节PID环时，Kp通常是其主要增益。
微分增益 –Kd (06)	如果误差或每10msPV就变化1个PV计数，这个INT数用来确定CV值的变化。输入代表10ms的低位时间，显示为0.00秒，有2位小数。例如，Kd输入120，将显示为1.20秒，结果是Kd*△误差/△时间，或，如果每30ms误差变化4PV计数，120*4/3作为PID输出的精度。Kd能用来提高低速回路的响应，但对PV输入噪音比较敏感。
积分速率 增益–Ki (07)	如果误差是一个常数PV计数，这个INT数用来确定CV值的变化。显示为0.000 重复/秒，有3位小数。例如，Ki输入1400，相识为1.400重复/秒，结果是Ki*误差或1400*20*50/1000，即误差为20PV计数，50msPLC扫描时间（采样周期为0）。Ki是Kp之后第二增益。
CV 偏置/输出 (08)	CV计数中一个INT 值加到PID输出，在斜率和放大限制前。如果只使用Kp比例增益，其可以设为0CV值，或为了从另一个控制环，形成这个PID环输出的前馈控制。

表 12-14. PID 参数详述 – 续

数据项	说 明
CV 上限和下限 (09/10)	CV数中的INT数定义CV的最高和最低数值。要求这些数值，上限必须比下限大，否则PID块将不工作。对于CV输出，这两个值通常按照物理限制为基础，用来定义限制。它们也用来测量LM90或ADS PID的CV的条形图。这个块有反对自设定终止功能，当达到CV限制时，修改积分器值。
最小回转时间 (11)	一个正数，定义CV输出从0到100%满量程或32000CV计数的最小时间。它与CV输出变化的斜率限制成反比。如果正，CV不能改变多于32000CV计数倍△时间（秒），△时间是最小回转时间的倒数。例如，如果采样周期是2.5秒，最小回转时间是500秒，CV不能改变多于32000*.5/500或每个PID计算160CV计数。由于由CV限制，有反对自设定终止特征，当达到CV限制时，修改积分器值。如果 回转时间为0，没有CV斜率限制。当你改变或调节PID环增益时，确定你设定最小回转时间为0。
定义字	这个字的低5位用来修改3个标准PID设定。其它位应该设定为0。 设定低位为1修改标准PID误差项，从平常的(SP – PV)到(PV – SP)，对反馈项的信号取反。这是为了反向控制，当PV超出，CV必须减小。设第2位为1，输出的极性取反，于是CV是PID输出的负数，而不是平常的正数。设定第4位为1，修改微分动作，从使用误差项内的平常改变到PV反馈项内的改变。 定义字中的低5位如下定义: **Bit 0** = 误差项。当这位设定为0，误差项为SP — PV。当设定为1，误差项为 PV — SP. **Bit 1** = 输出极性.当这位设定为0,CV输出表示PID计算的输出。当它设定为1，CV输出表示PID计算输出的负数。 **Bit 2** = PV上的微分动作。当这位设定为0，微分的动作作用于误差项。当设定为1，微分的动作作用于PV。所有保留位应该为0。 **Bit 3** = 死区动作。当死区动作位设定为0，就没有选择死区动作。如果误差在死区限制内，于是误差被强制为0。另外误差不受死区限制的影响。如果死区动作位设定为1，于是死区动作被选定。如果误差在死区限制内，于是误差被强制为0。如果，误差在死区限制外，于是误差减少死区限制（误差=误差-死区限制）。 **Bit 4** =反对自设定终止动作。当这位设定为0，反对自设定终止动作使用复位计算。当输出被限制时，这代替累积Y剩余值（12-87页定义），无论用什么值，必须产生正确的输出。当位设定为1，其用计算开始时Y项的值代替累积Y项。在这种方法中，予限制Y值同输出一起限制。 **注意:** 反对自设定终止动作位仅在6.50版或更后的 90-30 CPU。 记住，这些位在2电源中设定。例如，为了设定定义字为0，为默认PID配置，你将加1以改变误差项（从SP-PV到PV-SP），或加2以改变输出极性（从CV=PID输出到CV=-输出），或加4以改变微分动作（从改变的误差率到改变的PV斜率），等。

表 12-14. PID参数详述 – 续

数据项	说 明
手动命令 (13)	这是一个INT 数值，当PID块为自动模式时，设定当前CV输出。当块转变为手动模式时，在上限和下限内和回转时间限制内，这个值用来设定CV输出和积分器量的内部值。
控制字 (14)	这是个内部参数，通常为0。 如果不顾低位设定为1，这个字和其他内部SP，PV和CV参数必须用于PID块的远程操作（看下面）。这允许远程操作接口设备，如电脑，代替PLC程序控制。警告：如果你不想发生这些，确定使用控制字设定为0。如果低位为0，可以读下4位，跟踪PID输入接点（PID允许接点导通）的状态。前5位的离散数据结构如下形式： **位: 字 值: 功能: 状态或额外动作（如果不顾位设为1:** 0 1 不顾 如果为0，监视块连接下面。如果为1，额外设定它们。 1 2 手动/自动 如果为1，块为手动模式，其它数值块为自动模式。 2 4 允许 应该通常为1；其它情况， 块从不调用。 3 8 上/升 如果为1，手动（位1）为1，每个计算，CV递加。 4 16 下/降 如果为1，手动（位1）为1，每个计算，CV递减。
SP (15)	(没有可配置的设定和由PLC维持) 跟踪SP输入；如果不顾=1，必须额外设定。
CV (16)	(没有可配置的设定和由PLC维持) 跟踪CV输出。.
PV (17)	(没有可配置的设定和由PLC维持) 跟踪PV输入；如果不顾=1，必须额外设定。
输出(18)	(没有可配置的设定和由PLC维持) 这是一个有符号字数值，表示函数块的输出，在使用可选的取反前。如果没有定义输出取反，控制字中输出极性位设定为0，这个数值将等于CV输出。如果选择取反，输出极性位设定为1，这个数值将等于CV输出的负值。
微分项存储 (19)	内部使用来存储中间值。不写入这个位置。
积分项存储 (20/21)	内部使用来存储中间值。不写入这个位置。
回转项存储 (22)	内部使用来存储中间值。不写入这个位置。
时钟(23–25)	内部流逝时间存储(最近PID执行的时间)。不写入这个位置。
Y 剩余值 (26)	保留剩余值，作积分器换算的0稳态偏差。
下降和上升范围 (27/28)	PV计数内可选INT数值，其定义SP和PV最高和最低显示值，Logicmaster Zoom 水平条图和ADS PID平面显示。
保留的 (29–34 和35–39)	29–34 留着内部使用; 35–39 留着额外使用。它们留着 GE Fanuc使用，不能用作其它用途。

RefArray 内内部参数

如上页表12-3所述，PID块读13用户参数，使用其余的40字RefArray 用于内部PID存储。通常你不必改变任何这些值。如果在长延迟后，你用自动方式调用PID块，你可用SVC_REQ#16装载当前PLC流逝时间时钟进%Ref+23，更新最近PID解决时间，以避免积分器多级变化。如果你已经设定控制字（%Ref+14）的不顾低位为1，控制字的下四个位必须设定，控制PID块输入接点（如前页表12-3所述），当你已经把PID块从梯形图逻辑中撤出，内部SP和PV必须设定。

PID 运算法则选择(PIDISA 或 PIDIND) 和增益

PID块能选择独立(PID_IND)项或标准ISA (PID_ISA)两种PID算法之一进行编程。算法间唯一的不同是如何定义积分和微分增益。为了弄懂这些内容，你需要理解下面的东西：

两种PID类型计算误差项为SP-PV（相反动作），其可以变为直接动作模式（PV-SP），通过设定误差项为1。误差项是定义字（%Ref+0012）的低位（0位）。在直接动作比例（P）环中，处理变量（PV）的递增引起输出（CV）的递增。在取反动作比例（P）环中，处理变量（PV）的递增引起输出（CV）的递减。介绍的积分项（I）改变其运行。在直接动作PI环，当处理变量（PV）比设定值（SP）大时，输出（CV）将递增。在反向动作PI环，当处理变量（PV）比设定值（SP）大时，输出（CV）将递减。

直接动**作**:   误差 =测量值- 设定值 (PV-SP), 误差项 = 1

反向**动**作: 误差 =设定值- 测量值 (SP-PV), 误差项 = 0

注意.   直接动作有时称为前向动作。

微分通常基于误差项的改变，自从最近PID解决开始。如果SP数值改变了，其可能引起输出巨大变化。如果这不是希望的，定义字的第3位可以设定为1，基于PV的改变计算微分项。dt (或Δ时间) 由减去最近PID解决时钟决定，从当前PLC流逝时间减去。

dt = 当前PLC流逝时间 – PLC 流逝时间，最近PID解决

微分 = (误差 – 前一误差)/dt   或   (PV – 前一PV)/dt 如果定义字第3位设定为1。

独立项PID (PID_IND) 运算计算输出如: **PID 输出** =   Kp * 误差   +

Ki * 误差   * dt   +   Kd * 微分   +   CV 偏差

标准ISA (PID_ISA) 运算有不同形式:

PID 输出 =   Kc * (误差   +   误差   * dt/Ti   +   Td * 微分)   +   CV 偏差

Kc 是控制器增益，Ti是积分时间和Td是微分时间。ISA的好处是调整Kc改变积分和微分项的作用大小，如比例项一样，其可以使回路调节容易。如果你有PID增益项或Ti 和 Td, 用

Kp = Kc　　　　　　　　Ki = Kc/Ti　　　　　　和

Kd = Kc/Td ，以转变它们为PID用户参数输入。

上面的CV偏差项是一个附加项，从PID成分中分离开。当PV等于SP，误差为0时，你想CV是无0数值，如果你只使用比例增益Kp，可以要求CV偏差项。在这种情况，当PV等于PV，设定CV偏差为希望的CV。CV 偏差也可以用于前馈控制（另一个PID回路），或控制算法用来调节PID回路的CV输出。

如果使用积分Ki项，CV偏差通常是0，由于积分器值作为自动偏差。手动模式刚开始，使用自动命令字(%Ref+13)设定积分器值给希望的CV，于是转到自动模式。 如果Ki为0，其也工作，除非自动模式后，积分器值将不基于误差进行调节。

下图表示PID算法的运算:

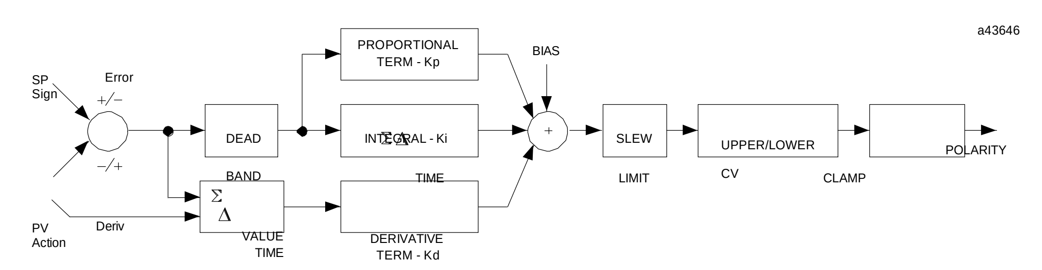


图 12-4. 独立项算法 (PIDIND)

ISA算法(PIDISA)类似，除了Kp 增益是Ki 和 Kd的外不因子，于是积分增益是Kp * Ki，微分增益是Kp * Kd。误差信号，微分和极性由用户参数定义字中的位设定。

CV 幅值和斜率

PID 块不会直接将计算得到的 PID 输出值赋给 CV。PID 的两种运算规则都会将振幅和变化率限制强制加在输出控制变量上。最大变化率等于 CV 最大值(32000)除以最小回转时间(如果最小回转时间大于 0)。例如，如果最小回转时间为 100 秒，转换率为每秒最大 320 个 CV 计数值。如果 dt 计算时间为 50 毫秒，那麼每次改变的最大变化量为 320*50/1000 或者说 16 个 CV 计数值。之后 CV 输出与上下限幅值比较。如果超过某个限幅值，则 CV 输出值设定为该限幅值。如果超过允许的比率或振幅，会调整内部积分器值来匹配限定值以避免 PID 重启终结

最后，PID 块会检查输出的极性(配置字%Ref+12 的第 2 位)，如果此位置 1，改变输出值的符号。

CV = PID 输出正限幅　　或者　如果输出极性被设定，为 PID 输出负限幅

如果PID块为自动模式，最后的CV值存放在手动命令字%Ref+13中。如果PID块为手动模式,CV值由手动命令设定，但是所有的比率和限幅值仍然起作用。这意味着，手动命令不能令CV值大于正限幅值，小于正限幅值，也不能令CV变化率大于最小回转时间设定的比率。

采样时间和PID时序

PID块是数字化的模拟量控制函数，所以PID输出方程式中的dt采样时间不能设为无穷小。受控的大部分过程可用一次或二次方程近似表示，相当于纯时间滞后。PID块设定一个CV输出给过程，并使用过程反馈回来的PV值来确定Error，以便调节下一次的CV输出值。总计时间是一个关键的过程参数，这个参数确定PV在多长时间后响应CV的变化。如下面关于设定回环增益部分所讨论的，在CV发生阶跃时，总计时间常量，Tp+Tc，是一阶系统PV值达到CV值63%所需要的时间。如果PID块的采样时间不小于总计时间的1/2，那麽PID块无法实现对此过程的控制。太大的采样时间会使控制不稳定 。

采样时将不应大于总计时间的1/10(最差的情况下也不能大于总计时间的1/5)。例如，如果预计PV值会用2秒钟的时间到达最终值的2/3，采样时间应小于0.2秒，最差的情况下也不能大于0.4秒。另一方面，采样时间也不能太小，例如小于总计时间的1/1000，或者积分器的Ki * Error * dt值趋向于0的情况。例如，如果有这样一个非常缓慢的过程，预计PV值会用10小时或者36000秒钟的时间到达最终值的63%时，采样时间应大于或等于40秒。

除非过程非常快，否则不需要将采样时间设为0，从而每个扫描周期都进行PID运算。如果很多PID回路使用了大于扫描周期的采样时间，那麽在有很多PID环同时完成计算时，CPU的扫描时间会有很大的变化。解决的办法是将控制PID块运行的位排在一个数列内，之后按顺序的把数列内的一位或几位数据置0。

确定过程参数

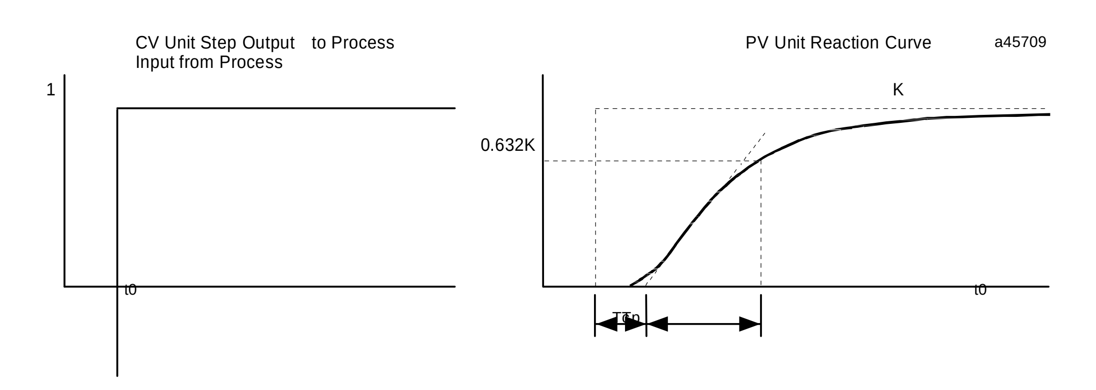
PID 回路增益，Kp, Ki 和 Kd,由所控制的过程特征决定。建立PID回路两个关键问题为：

1. CV改变一定量PV改变有多大，或者开路增益是什么？
2. 系统响应有多快，或者CV输出阶跃后PV改变有多快？

许多过程可以近似为过程增益，第1和第2阶导数和纯延时。在频域内，带纯延时的一阶系统的传递函数是s:

PV(s)/CV(s) = G(s) = K * e **(–Tp s)/(1 + Tc s)

时域内t0时刻的开环单位阶跃响应曲线如下:



以下过程模型参数由PV单位作用曲线得到:

K	过程开路增益 = PV最终改变/t0时刻CV改变 (注意K无脚标)
Tp	t0之后输出PV开始之前过程延迟时间或死区时间
Tc	第一个时间常数之后, Tp之后 PV 达到最终PV值的 63.2%所需时间。

通常测量这些参数最快的方法就是把PID模块置于手动方式并把CV输出作较小阶跃，改变Manual Command %Ref+13, 并且绘制 PV 响应时间曲线。对于慢速过程，该工作可以用手工方式完成，但对于快速过程，借助图形记录仪或计算机图形数据记录包。CV阶跃幅度应足够引起PV的可观察变化，但也不要太大以至于影响所测量的过程。最佳范围为CV上下限之差的2 ～ 10% 。

设置用户参数包括调节回路增益

由于所有 PID 参数取决于所控制的过程，所以没有现成的预定值，但是很容易找到合适的回路增益。

1. 设置所有用户参数为 0, 设置 CV 上下限为最高和最低 CV 值。设定采样周期为预定的过程时间常数 10 到 100.
2. 把模块置于手动模式，并在不同值下设置 Manual Command (%Ref+13) 以检查 CV 是否可以达到上下限。记录 CV 点的 PV 值并装入 SP。
3. 设定较小增益，比如 100 * 最大 CV/最大 PV, 为 Kp 并且中止手动模式。使 SP 值在 PV 最大范围的 2 ～ 10% 变化，并且观察 PV 响应情况。如果 PV 响应太慢则增大 Kp，若 PV 溢出并且振荡 则减小 Kp。
4. 一旦找到 Kp 后，开始增加 Ki 直到溢出，该溢出应在 2～3 个循环中减弱至稳定值。这可以通过减小 Kp 实现。也可以试试不同的阶跃和 CV 运行点。
5. 找到稳定的 Kp 和 Ki 增益后, 试试增加 Kd 以获得较快的输入响应且不产生振荡。Kd 通常不需要而且有干扰 PV 时无法工作。
6. 检查不同SP操作点时的增益，必要时加上死区和最小回转时间。有些反向过程还要求设置配置字误差标志位或极性位。

设置回路增益－ Ziegler 和 Nichols 调节方法

一旦K, Tp 和 Tc三个过程模型参数确定后，即可用来评估初始PID回路增益。以下方法，由Ziegler 和 Nichols在20世纪40年代提出，能很好相应系统干扰，产生的幅比为1/4. 幅比即闭环回路第二个峰值与第一个峰值之比。

1. 计算作用速率:

   R = K/Tc

2. 只对比例控制，计算 Kp :

   Kp = 1/(R * Tp) = Tc/(K * Tp)

3. 对比例和积分控制, 如下计算:

   Kp = 0.9/(R * Tp) = 0.9 * Tc/(K * Tp) Ki = 0.3 * Kp/Tp

4. 对于比例，积分，和微分控制, 如下计算:

   Kp = G/(R * Tp) where G is from 1.2 to 2.0
   Ki = 0.5 * Kp/Tp
   Kd = 0.5 * Kp * Tp

5. 确认采样周期在以下范围：

   (Tp + Tc)/10 to (Tp + Tc)/1000

另一种方法，理想过程，理想调节方法仅通过 Tp 过程延迟或死区时间延迟可获得最佳的 SP 改变响应。

Kp = 2 * Tc/(3 * K * Tp)
Ki = Tc
Kd = Ki/4 如果使用微分项

一旦确定初始增益, 将其转换为整数。按照输入 PV 值的改变除以 CV 值的输出跃变改变计算过程增益 K ，而不要使用 PV 或 CV 工程单位计算。所有时间用秒作为单位。一旦 Kp, Ki 和 Kd 确定后，Kp 和 Kd 可以乘以 100 并且按整数输入，Ki 可以乘以 1000 并输入到用户参数%RefArray 中去.

PID调用举例

以下PID 采样周期为 100ms, Kp参数为4.00 ，Ki 参数为1.500. 设定存储在地址%R0001中, 控制输出为 %AQ0002, 反馈值为 %AI0003. CV 上下限必须设定这里设为 20000 和 400, 并且设定了死区Dead Band 为+5 和 -5 。长度为40个字的参数块起始地址为%R0100. 闭合接点%M0006，允许一对BLKMV指令，其通过复制常数到从%R102 (%Ref+2)开始的14个字中，以设定初始参数值。（注意：在调整过程中，为了使参数最优，通常用户参数用PID ZOOM键F10进入）。本块能用%M0001切换到手动模式，于是手动命令，%R0113，能调节。位%M0004或M0005用来递增或递减%R0113，PID CV和100ms积分一次。对于更快的手动操作，位%M0002和%M0003用来增加或减少%R0113，每个扫描变化量为%R0002数值。当PID OK时，%T0001输出为ON。注意40个寄存器参数块的一些寄存器不包括在内，因为本例子中没有使用它们，或他们没有被配置，因为PLC系统使用它们。关于附加的参数信息，看表 12-8.

地址	值	说明
%R0102	+00010	采用周期
%R0103	+00005	死区 +
%R0104	+00005	死区 ↕
%R0105	+00400	比例增益 (Kp)
%R0106	+00000	微分增益 (Kd)
%R0107	+01500	积分增益 (Ki)
%R0108	+00000	CV偏差/输出偏移
%R0109	+20000	上限
%R0110	+00400	下限
%R0111	+00000	最小回转时间
%R0112	+00000	定义字
%R0113	+00000	手动命令
%R0114	+00000	控制字
%R0115	+00000	内部 SP (不可配置)

```
|           _____          _____              _____
| %M0006 |       |        |     |            |     |
|--| |---| BLK_  |--------|BLKMV|------------|BLKMV|-
|        |       |        |     |            |     |
|        | CLR_  |        | INT |            | INT |
|        | WORD  |        |     |            |     |
|%R00100-|IN     | CONST -|IN1 Q|-%R00102 CONST -|IN1 Q|- %R00109
|        | LEN   | +00010 |     |         +20000 |     |
|        |00035  |        |     |                |     |
|        |_______| CONST -|IN2  |          CONST -|IN2  |
|                  +00005 |     |          +00400 |     |
|                         |     |                 |     |
|                  CONST -|IN3  |          CONST -|IN3  |
|                  +00005 |     |          +00000 |     |
|                         |     |                 |     |
|                  CONST -|IN4  |          CONST -|IN4  |
|                  +00400 |     |          +00000 |     |
|                         |     |                 |     |
|                  CONST -|IN5  |          CONST -|IN5  |
|                  +00000 |     |          +00000 |     |
|                         |     |                 |     |
|                  CONST -|IN6  |          CONST -|IN6  |
|                  +01500 |     |          +00000 |     |
|                         |     |                 |     |
|                  CONST -|IN7  |          CONST -|IN7  |
|                  +00000 |_____|          +00000 |_____|
|
|                                          _____
|ALW_ON                                   |     | %T0001
|--| |------------------------------------| PID_|--( )--
|                                         | IND |
|                                         |     |
|                                %R0001---|SP CV|- %AQ002
|                                         |     |
|                                         |     |
|                               %AI0003---|PV   |
|                                         |     |
|%M0001                                   |     |
|--| |------------------------------------|MAN  |
|                               |          |     |
|                               | %M0004   |     |
|                               |--| |-----|UP   |
|                               |          |     |
|                               | %M0005   |     |
|                               --| |------|DN   |
|                                          |_____|
|
|                                          %R00100
|          _____
|%M0002  |       |
|--| |---| ADD_  |----
|        | INT   |
|        |       |
|%R00113-|I1   Q|- %R00113
|        |       |
|        |       |
| %R0002-|I2     |
|        |       |
|        |_______|
|
|          _____
|%M0003  |       |
|--| |---| SUB_  |-
|        | INT   |
|        |       |
|%R00113-|I1   Q|- %R00113
|        |       |
| %R0002-|I2     |
|        |       |
|        |_______|
```

附录 A 指令的分时

系列90-30，90-20和Micro PLC将支持多种不同的功能和功能块。这一附录中将包含一些表，其中列出了每种功能模块用字节表示的存贮器容量及用微秒表示的时间。存贮器容量是由梯形图应用程序中功能模块所需的字节数表示。
对每种功能模块来说，两种执行时间如下所示：

执行时间	说明
使能	当使能并流出电流时，执行功能或功能模块所需的时间，特别是，当所用数据块在用户RAM（字存贮器）中，而不是在数字存贮器中，为最佳情况。
禁止	当功能或功能模块流入电流时，执行功能所需时间，但是，当定时器处于复位状态时，它将处于一不作用的状态。

注意

定时器和计数器当其每次接入逻辑中便被更新，定时器由最后一次扫描所耗时间量更新，计数器由一次计数更新。

注意

对于 350, 351, 352, 和 360 PLC的 CPU来说,指令执行时间都一样，除了350 CPU的MOVE指令—参见A-6表底部的注意.

Table A-1. 指令的分时,标准模块

功能分类	功能模块	允许 311	允许 313	允许 331	允许 340/41	禁止 311	禁止 313	禁止 331	禁止 340/41	增量 311	增量 313	增量 331	增量 340/41	容量
定时器	接通延时定时器	146	81	80	42	105	39	38	21	–	–	–	–	15
	断开延时定时器	98	47	44	23	116	63	58	32	–	–	–	–	9
	定时器	122	76	75	40	103	54	53	30	–	–	–	–	15
计数器	加计数器	137	70	69	36	130	63	62	33	–	–	–	–	11
	减计数器	136	70	69	37	127	61	61	31	–	–	–	–	11
数学运算	加 (INT)	76	47	46	24	41	0	1	0	–	–	–	–	13
	加(DINT)	90	60	60	34	41	1	0	0	–	–	–	–	13
	减 (INT)	75	46	45	25	41	0	1	0	–	–	–	–	13
	减(DINT)	92	62	62	34	41	1	0	0	–	–	–	–	13
	乘 (INT)	79	49	50	28	41	0	1	0	–	–	–	–	13
	乘 (DINT)	108	80	101	43	41	1	0	0	–	–	–	–	13
	除 (INT)	79	51	50	27	41	0	1	0	–	–	–	–	13
	除 (DINT)	375	346	348	175	41	1	0	0	–	–	–	–	13
	模除 (INT)	78	51	49	27	41	0	1	0	–	–	–	–	13
	模除(DINT)	134	103	107	54	41	1	0	0	–	–	–	–	13
	平方根t (INT)	153	124	123	65	42	0	1	0	–	–	–	–	9
	平方根 (DINT)	268	239	241	120	42	0	0	1	–	–	–	–	9
关系运算	等于 (INT)	66	35	36	19	41	1	1	0	–	–	–	–	9
	等于 (DINT)	86	56	54	29	41	1	0	0	–	–	–	–	9
	不等于 (INT)	67	39	35	22	41	1	1	0	–	–	–	–	9
	不等于 (DINT)	81	51	51	28	41	1	0	0	–	–	–	–	9
	大于 (INT)	64	33	35	20	41	1	1	0	–	–	–	–	9
	大于 (DINT)	89	59	58	32	41	1	0	0	–	–	–	–	9
	大于/等于 (INT)	64	36	34	19	41	1	1	0	–	–	–	–	9
	大于/等于 (DINT)	87	58	57	30	41	1	0	0	–	–	–	–	9
	小于 (INT)	66	35		19	41	1	1	0	–	–	–	–	9
	小于 (DINT)	87	57		30	41	1	1	0	–	–	–	–	9
	小于/等于 (INT)	66	36	34	21	41	1	1	0	–	–	–	–	9
	小于/等于 (DINT)	86	57	56	31	41	1	1	0	–	–	–	–	9
	范围 (INT)	92	58	54	29	46	1	0	1	–	–	–	–	15
	范围(DINT)	106	75	57	37	45	0	0	0	–	–	–	–	15
	范围(WORD)	93	60	54	29	0	0	0	0	–	–	–	–	15

注意:
1. 对于制表功能, 增量指定单元长度.; 低于位运算功能, 微秒.; 对于数据移动功能, 微秒/位数或字数.
2. 对于单独的单元长度类型 %R, %AI, 和 %AQ.
3. COMMREQ 使用 CPU 和 HSC测试.
4. DOIO为输出到离散输出模块时间 .
5. 有多种情况下,以上时间表示最坏情况下.
6. 对于含有增量的指令, (长度 –1)乘以增量时间加到基本时间上

Table A-1.　指令的分时,标准模块

功能分类	功能模块	允许				禁止				增量				容量
		311	313	331	340/41	311	313	331	340/41	311	313	331	340/41	
位操作	逻辑与 AND	67	37	37	22	42	0	0	1	–	–	–	–	13
	逻辑或 OR	68	38	38	21	42	0	0	1	–	–	–	–	13
	逻辑异或 OR	66	38	37	20	42	0	1	1	–	–	–	–	13
	逻辑非 NOT	62	32	31	17	42	0	1	1	–	–	–	–	9
	左移位	139	89	90	47	74	26	23	13	11.61	11.61	12.04	6.29	15
	右移位	135	87	85	45	75	26	24	13	11.63	11.62	12.02	6.33	15
	循环左移	156	127	126	65	42	1	1	0	11.70	11.78	12.17	6.33	15
	循环右移	146	116	116	62	42	1	1	0	11.74	11.74	12.13	6.27	15
	定位	102	72	49	38	42	1	0	0	–	–	–	–	13
	位清除	68	38	35	21	42	1	1	1	–	–	–	–	13
	位检测	79	49	51	28	41	0	0	1	–	–	–	–	13
	置位	67	37	37	20	42	0	0	0	–	–	–	–	13
	屏蔽比较 (WORD)	217	154	141	74	107	44	39	21	–	–	–	–	25
	屏蔽比较 (DWORD)	232	169	156	83	108	44	39	22	–	–	–	–	25
数据传送	传送 (INT)	68	37	39	20	43	0	0	0	1.62	1.62	5.25	1.31	13
	传送 (BIT)	94	62	64	35	42	0	0	0	12.61	12.64	12.59	6.33	13
	传送 (WORD)	67	37	40	20	41	0	0	0	1.62	1.63	5.25	1.31	13
	块传送 (INT)	76	48	50	28	59	30	30	16	–	–	–	–	27
	块传送(WORD)	76	48	49	29	59	29	28	15	–	–	–	–	27
	块清除	56	28	27	14	43	0	0	0	1.35	1.29	1.40	0.78	9
	移位寄存器 (BIT)	201	153	153	79	85	36	34	18	0.69	0.68	0.71	0.37	15
	移位寄存器 (WORD)	103	53	52	29	73	25	23	12	1.62	1.62	2.03	1.31	15
	位定序器	165	101	99	53	96	31	29	16	0.07	0.07	0.08	0.05	15
	通讯请求	1317	1272	1489	884	41	2	0	0	–	–	–	–	13
功能制表	数组传送													
	INT	230	201	177	104	72	41	40	20	1.29	1.15	10.56	2.06	21
	DINT	231	202	181	105	74	44	42	23	3.24	3.24	10.53	2.61	21
	BIT	290	261	229	135	74	43	42	23	–.03	–.03	-0.01	0.79	21
	BYTE	228	198	176	104	74	42	42	23	0.81	0.82	8.51	1.25	21
	WORD	230	201	177	104	72	41	40	20	1.29	1.15	10.56	2.06	21
	查寻相等													
	INT	197	158	123	82	78	39	37	20	1.93	1.97	2.55	1.55	19
	DINT	206	166	135	87	79	38	36	21	4.33	4.34	4.55	2.44	19
	BYTE	179	141	117	74	78	38	36	21	1.53	1.49	1.83	1.03	19
	WORD	197	158	123	82	78	39	37	20	1.93	1.97	2.55	1.55	19

注意:
1. 对于制表功能, 增量指定单元长度.; 低于位运算功能, 微秒.; 对于数据移动功能, 微秒/位数或字数.
2. 对于单独的单元长度类型 %R, %AI, 和 %AQ.
3. COMMREQ 使用 CPU 和 HSC测试.
4. DOIO为输出到离散输出模块时间 .
5. 有多种情况下,以上时间表示最坏情况下.
6. 对于含有增量的指令, (长度 –1)乘以增量时间加到基本时间上

Table A-1. 指令的分时,标准模块-续

模块分类	功能模块	允许				禁止				增量				容量
		311	313	331	340/41	311	313	331	340/41	311	313	331	340/41	
	查寻不相等													
	INT	198	159	124	83	79	39	36	21	1.93	1.93	2.48	1.52	19
	DINT	201	163	132	84	79	37	35	21	6.49	6.47	6.88	3.82	19
	BYTE	179	141	117	73	79	38	36	19	1.54	1.51	1.85	1.05	19
	WORD	198	159	124	83	79	39	36	21	1.93	1.93	2.48	1.52	19
	查寻大于													
	INT	198	160	125	82	79	37	38	19	3.83	3.83	4.41	2.59	19
	DINT	206	167	135	88	78	38	36	20	8.61	8.61	9.03	4.88	19
	BYTE	181	143	118	73	79	37	36	19	3.44	3.44	3.75	2.03	19
	WORD	198	160	125	82	79	37	38	19	3.83	3.83	4.41	2.59	19
	查寻大于/等于													
	INT	197	160	124	83	77	38	36	20	3.86	3.83	4.45	2.52	19
	DINT	205	167	136	87	80	39	36	21	8.62	8.61	9.02	4.87	19
	BYTE	180	142	118	75	79	37	37	20	3.47	3.44	3.73	2.00	19
	WORD	197	160	124	83	77	38	36	20	3.86	3.83	4.45	2.52	19
	查寻小于													
	INT	199	159	124	84	78	38	36	20	3.83	3.86	4.48	2.48	19
	DINT	206	168	135	87	79	38	38	19	8.62	8.60	-1.36	4.88	19
	BYTE	181	143	119	75	80	38	37	20	3.44	3.44	3.75	2.00	19
	WORD	199	159	124	84	78	38	36	20	3.83	3.86	4.45	2.48	19
	查寻小于/等于													
	INT	200	158	124	82	79	38	37	21	3.79	3.90	4.45	2.55	19
	DINT	207	167	137	88	78	39	37	19	8.60	8.61	9.01	4.86	19
	BYTE	180	143	119	74	78	40	37	19	3.46	3.44	3.73	2.02	19
	WORD	200	158	124	82	79	38	37	21	3.79	3.90	4.45	2.55	19
转换功能	转换到INT	74	46	39	25	42	1	1	1	–	–	–	–	9
	转换到BCD–4	77	50	34	25	42	1	1	1	–	–	–	–	9

注意:
1. 对于制表功能, 增量指定单元长度.; 低于位运算功能, 微秒.; 对于数据移动功能, 微秒/位数或字数.
2. 对于单独的单元长度类型 %R, %AI, 和 %AQ.
3. COMMREQ 使用 CPU 和 HSC测试.
4. DOIO为输出到离散输出模块时间 .
5. 有多种情况下,以上时间表示最坏情况下.
6. 对于含有增量的指令, (长度 –1)乘以增量时间加到基本时间上.

Table A-1. 指令的分时,标准模块-续

功能分类	功能模块	允许				禁止				增量				容量
		311	313	331	340/41	311	313	331	340/41	311	313	331	340/41	
控制功能	调用一个子程序	155	93	192	85	41	0	0	0	–	–	–	–	7
	Do I/O	309	278	323	177	38	1	0	0	–	–	–	–	12
	PID – ISA算法	1870	1827	1812	929	91	56	82	30	–	–	–	–	15
	PID – IND 算法	2047	2007	2002	1017	91	56	82	30	–	–	–	–	15
	End 指令	–	–	–	–	–	–	–	–	–	–	–	–	–
	服务请求													
	# 6	93	54	63	45	41	2	0	0	–	–	–	–	9
	# 7 (Read)	–	37	309	161	–	2	0	0	–	–	–	–	9
	# 7 (Set)	–	37	309	161	–	2	0	0	–	–	–	–	9
	#14	447	418	483	244	41	2	0	0	–	–	–	–	9
	#15	281	243	165	139	41	2	0	0	–	–	–	–	9
	#16	131	104	115	69	41	2	0	0	–	–	–	–	9
	#18	–	56	300	180	–	2	0	0	–	–	–	–	9
	#23	1689	1663	1591	939	43	1	0	0	–	–	–	–	9
	#26//30*	1268	1354	6680	3538	42	0	0	0	–	–	–	–	9
	#29	–	–	55	41	–	–	1	0	–	–	–	–	9
	嵌套 MCR/ENDMCR	135	73	68	39	75	25	21	12	–	–	–	–	8

**SVCREQ #26/30 使用高速计数器，16点输出，5槽机架测试。

注意:
1. 对于制表功能, 增量指定单元长度.; 低于位运算功能, 微秒.; 对于数据移动功能, 微秒/位数或字数.
2. 对于单独的单元长度类型 %R, %AI, 和 %AQ.
3. COMMREQ 使用 CPU 和 HSC测试.
4. DOIO为输出到离散输出模块时间 .
5. 有多种情况下,以上时间表示最坏情况下.
6. 对于含有增量的指令, (长度 –1)乘以增量时间加到基本时间上.

Table A-2. 指令的分时, 35x-36x 模板

功能分类	功能模块	允许 350/351/36x	禁止 350/351/36x	增量 350/351/36x	允许 352	禁止 352	增量 352	容量
定时器	接通延时定时器	4	6	–	4	5	–	15
	定时器	3	3	–	2	2	–	15
	断开延时定时器	3	3	–	3	2	–	15
计数器	加计数器	1	3	–	2	2	–	13
	减计数器	3	3	–	1	2	–	13
数学运算	加 (INT)	2	0	–	1	0	–	13
	加(DINT)	2	0	–	2	0	–	19
	加 (REAL)	52	0	–	33	0	–	17
	减(INT)	2	0	–	1	0	–	13
	减(DINT)	2	0	–	2	0	–	19
	减 (REAL)	53	0	–	34	0	–	17
	乘 (INT)	21	0	–	21	0	–	13
	乘 (DINT)	24	0	–	24	0	–	19
	乘 (REAL)	68	1	–	38	1	–	17
	除 (INT)	22	0	–	22	0	–	13
	除 (DINT),	25	0	–	25	0	–	19
	除 (REAL)	82	2	–	36	2	–	17
	模除 (INT)	21	0	–	21	0	–	13
	模除 (DINT)	25	0	–	25	0	–	19
	平方根 (INT)	42	1	–	41	1	–	10
	平方根 (DINT)	70	0	–	70	0	–	13
	平方根 (REAL)	137	0	–	35	0	–	11
三角函数运算	SIN (REAL)	360	0	–	32	0	–	11
	COS (REAL)	319	0	–	29	0	–	11
	TAN (REAL)	510	1	–	32	1	–	11
	ASIN (REAL)	440	0	–	45	0	–	11
	ACOS (REAL)	683	0	–	63	0	–	11
	ATAN (REAL)	264	1	–	33	1	–	11
对数运算	LOG (REAL)	469	0	–	32	0	–	11
	LN (REAL)	437	0	–	32	0	–	11
指数运算	EXP	639	0	–	42	0	–	11
	EXPT	89	1	–	54	1	–	17
弧度转换	转换RAD 到 DEG	65	1	–	32	1	–	11
	转换DEG 到RAD	59	0		32	0		11

注意:
1. 对于制表功能，增量指定单元长度.; 低于位运算功能，微秒.; 对于数据移动功能，微秒/位数或字数.
2 对于单独的单元长度类型 %R, %AI, 和 %AQ.
3. COMMREQ 使用 CPU 和 HSC测试.
4. DOIO为输出到离散输出模块时间 .
5. 有多种情况下,以上时间表示最坏情况下.
6. 对于含有增量的指令, (长度 –1)乘以增量时间加到基本时间上.

Table A-2. 指令的分时, 35x-36x 模板-续

功能分类	功能模块	允许 350/351/36x	禁止 350/351/36x	增量 350/351/36x	允许 352	禁止 352	增量 352	容量
关系运算	等于 (INT)	1	0	–	1	0	–	10
	等于 (DINT)	2	0	–	2	0	–	16
	等于 (REAL)	57	0	–	28	0	–	14
	不等于(INT)	1	0	–	1	0	–	10
	不等于 (DINT)	1	0	–	1	0	–	16
	不等于 (REAL)	62	0	–	31	0	–	14
	大于 (INT)	1	0	–	1	0	–	10
	大于 (DINT)	1	0	–	1	0	–	16
	大于 (REAL)	57	0	–	32	0	–	14
	大于/等于 (INT)	1	0	–	1	0	–	10
	大于/等于 (DINT)	1	0	–	1	0	–	10
	大于/等于(REAL)	57	1	–	31	1	–	14
	小于 (INT)	1	0	–	1	0	–	10
	小于(DINT)	1	0	–	1	0	–	16
	小于(REAL)	58	1	–	36	1	–	14
	小于/等于 (INT)	1	0	–	1	0	–	10
	小于/等于(DINT)	3	0	–	3	0	–	16
	小于/等于(REAL)	37	0	–	37	0	–	14
	范围 (INT)	2	1	–	2	1	–	13
	范围(DINT)	2	1	–	2	1	–	22
	范围(WORD)	1	0	–	1	0	–	13
位操作	逻辑与AND	2	0	–	2	0	–	13
	逻辑或OR	2	0	–	2	0	–	13
	逻辑异或OR	1	0	–	1	0	–	13
	逻辑非, NOT	1	0	–	1	0	–	10
	左移位	31	1	1.37	31	1	1.37	16
	右移位	28	0	3.03	28	0	3.03	16
	循环左移	25	0	3.12	25	0	3.12	16
	循环右移	25	0	4.14	25	0	4.14	16
	定位	20	1	–	20	1	–	13
	位清除	20	0	–	20	0	–	13
	位检测	20	0	–	20	0	–	13
	置位	19	1	–	19	1	–	13
	屏蔽比较(WORD)	52	0	–	52	0	–	25
	屏蔽比较(DWORD)	50	0	–	49	0	–	25

注意:
1. 对于制表功能，增量指定单元长度.; 低于位运算功能，微秒.; 对于数据移动功能，微秒/位数或字数.
2. 对于单独的单元长度类型 %R, %AI, 和 %AQ.
3. COMMREQ 使用 CPU 和 HSC测试.
4. DOIO为输出到离散输出模块时间 .
5. 有多种情况下,以上时间表示最坏情况下.
6. 对于含有增量的指令, (长度 –1)乘以增量时间加到基本时间上.

Table A-2. 指令的分时, 35x-36x 模板-续

功能分类	功能模块	允许 350/351/36X	禁止 350/351/36X	增量 350/351/36X	允许 352	禁止 352	增量 352	容量
数据传送	传送 (INT)	2	0	0.41	2	0	0.41	10
	传送 (BIT)	28	0	4.98	28	0	4.98	13
	传送 (WORD)	2	0	0.41	2	0	0.41	10
	传送 (REAL)	24	1	0.82	24	1	0.82	13
	块传送 (INT)	2	0	–	2	0	–	28
	块传送(WORD)	4	4	–	3	0	–	28
	块传送(REAL)	41	0	–	41	0	–	13
	块清除	1	0	0.24	1	0	0.24	11
	移位寄存器 (BIT)	49	0	0.23	46	0	0.23	16
	移位寄存器(WORD)	27	0	0.41	27	0	0.41	16
	位定序器	38	22	0.02	38	22	0.02	16
	通讯请求	765	0	–	765	0	–	13
功能表	数组传送							
	INT	54	0	0.97	54	0	0.97	22
	DINT	54	0	0.81	54	0	0.81	22
	BIT	69	0	0.36	69	0	0.36	22
	BYTE	54	1	0.64	54	1	0.64	22
	WORD	54	0	0.97	54	0	0.97	22
	查寻相等							
	INT	37	0	0.62	37	0	0.62	19
	DINT	41	1	1.38	41	1	1.38	22
	BYTE	35	0	0.46	35	0	0.46	19
	WORD	37	0	0.62	37	0	0.62	19
	查寻不相等							
	INT	37	0	0.62	37	0	0.62	19
	DINT	38	0	2.14	38	0	2.14	22
	BYTE	37	0	0.47	37	0	0.47	19
	WORD	37	0	0.62	37	0	0.62	19
	查寻大于							
	INT	37	0	1.52	37	0	1.52	19
	DINT	39	0	2.26	39	0	2.26	22
	BYTE	36	1	1.24	36	1	1.24	19
	WORD	37	0	1.52	37	0	1.52	19
	查寻大于/等于							
	INT	37	0	1.48	37	0	1.48	19
	DINT	39	0	2.33	39	0	2.33	22
	BYTE	37	1	1.34	37	1	1.34	19
	WORD	37	0	1.48	37	0	1.48	19

注意:
1. 对于制表功能, 增量指定单元长度.; 低于位运算功能, 微秒.; 对于数据移动功能, 微秒/位数或字数.
2. 对于单独的单元长度类型 %R, %AI, 和 %AQ.
3. COMMREQ 使用 CPU 和 HSC测试.
4. DOIO为输出到离散输出模块时间 .
5. 有多种情况下,以上时间表示最坏情况下.
6. 对于含有增量的指令, (长度 –1)乘以增量时间加到基本时间上.

Table A-2. 指令的分时, 35x-36x 模板-续

功能分类	功能模块	允许 350/351/36x	禁止 350/351/36x	增领 350/351/36x	允许 352	禁止 352	增量 352	容量
	查寻小于							
	INT	37	0	1.52	37	0	1.52	19
	DINT	41	1	2.27	41	1	2.27	22
	BYTE	37	0	1.41	37	0	1.41	19
	WORD	37	0	1.52	37	0	1.52	19
	查寻小于/等于							
	INT	38	0	1.48	38	0	1.48	19
	DINT	40	1	2.30	40	1	2.30	22
	BYTE	37	0	1.24	37	0	1.24	19
	WORD	38	0	1.48	38	0	1.48	19
转换功能	转换成 INT	19	1	–	19	1	–	10
	转换成 BCD-4	21	1	–	21	1	–	10
	转换成 REAL	27	0	–	21	0	–	8
	转换成 WORD	28	1	–	30	1	–	11
	舍位到 INT	32	0	–	32	0	–	11
	舍位到 DINT	63	0	–	31	0	–	11
控制功能	调用一个子程序	72	1	–	73	1	–	7
	Do I/O	114	1	–	115	1	–	13
	PID – ISA 算法*	162	34	–	162	34	–	16
	PID – IND 算法*	146	34	–	146	34	–	16
	End 指令	–	–	–	–	–	–	–
	服务请求							
	#6	22	1	–	22	1	–	10
	#7 (Read)	75	1	–	75	1	–	10
	#7 (Set)	75	1	–	75	1	–	10
	#14	121	1	–	121	1	–	10
	#15	46	1	–	46	1	–	10
	#16	36	1	–	36	1	–	10
	#18	261	1	–	261	1	–	10
	#23	426	0	–	426	0	–	10
	#26//30**	2260	1	–	2260	1	–	10
	#29	20	0	–	20	0	–	10
	#43							
	嵌套 MCR/ENDMCR 嵌套	1	1	–	1	1	–	4
	SER	见表 A-3	26.50	见表 A-3				

***** PID 功能时间基于CPU351 6.5 版本.

******SVCREQ #26/30 使用高速计数器，16点输出，5槽机架测试。

注意:
1. 对于制表功能, 增量指定单元长度.; 低于位运算功能, 微秒.; 对于数据移动功能, 微秒/位数或字数.
2. 对于单独的单元长度类型 %R, %AI, 和 %AQ.
3. COMMREQ 使用 CPU 和 HSC测试.
4. DOIO为输出到离散输出模块时间 .
5. 有多种情况下,以上时间表示最坏情况下.
6. 对于含有增量的指令, (长度 –1)乘以增量时间加到基本时间上.

表 A-3. SER 功能块

配置	示例	时间 (μsec)
无电流 (禁止)	—	26.50
Contiguous		
8 通道	%I1—8	79.94
16通道	%I1—16	80.58
24通道	%I1—24	81.56
32通道	%I1—32	81.73
8 + 8 邻近通道	%I1—8 and %Q1—8	111.03
8 + 8 + 8 邻近通道	%I1—8, %Q1—8 和 %M1—8	143.38
8 + 8 + 8 + 8 邻近通道	%I1—8, %Q1—8 and %M1—8和 %T1—8	175.79
Noncontiguous	%I1, %M10, %Q3, 等.	
8通道		299.64
16通道		552.83
24通道		806.35
32通道		1059.85
Reset		
8 通道	—	162.63
16通道	—	267.51
24通道	—	372.73
32通道	—	477.95

注意: 如果该槽是输入模块，那么每个**Contiguous** 和 **Noncontiguous**增加46 μsecs.

触发发生, 如果使用 BCD 格式增加29 usec或如果使用Posix格式增加148 usec.

复位是最大缓冲器1024 . (复位清除缓冲器所有采样)

Table A-4. 指令的分时, 37x 模板

功能分类	功能模块	允许 37x	禁止 37x	增量 37x	容量
定时器	接通延时定时器	4	5	–	15
	定时器	2	2	–	15
	断开延时定时器	3	2	–	15
计数器	加计数器	2	2	–	13
	减计数器	1	2	–	13
数学运算	加 (INT)	1	0	–	13
	加(DINT)	2	0	–	19
	加 (REAL)	5	0	–	17
	减(INT)	1	0	–	13
	减 (DINT)	2	0	–	19
	减(REAL)	5	0	–	17
	乘 (INT)	5	0	–	13
	乘 (DINT)	5	0	–	19
	乘 (REAL)	5	0	–	17
	除(INT)	5	0	–	13
	除 (DINT),	5	0	–	19
	除 (REAL)	5	0	–	17
	模除 (INT)	5	0	–	13
	模除 (DINT)	5	0	–	19
	平方根 (INT)	5	0	–	10
	平方根 (DINT)	10	0	–	13
	平方根 (REAL)	5	0	–	11
三角函数运算	SIN (REAL)	10	0	–	11
	COS (REAL)	10	0	–	11
	TAN (REAL)	10	0	–	11
	ASIN (REAL)	10	0	–	11
	ACOS (REAL)	10	0	–	11
	ATAN (REAL)	5	0	–	11
对数运算	LOG (REAL)	5	0	–	11
	LN (REAL)	5	0	–	11
指数运算	EXP	10	0	–	11
	EXPT	10	0	–	17
弧度转换	转换 RAD 到 DEG	5	0	–	11
	转换 DEG 到 RAD	5	0		11

注意:
1. 对于制表功能，增量指定单元长度.; 低于位运算功能，微秒.; 对于数据移动功能，微秒/位数或字数.
2. 对于单独的单元长度类型 %R, %AI, 和 %AQ.
3. COMMREQ 使用 CPU 和 HSC测试.
4. DOIO为输出到离散输出模块时间 .
5. 有多种情况下,以上时间表示最坏情况下.
6. 对于含有增量的指令, (长度 –1)乘以增量时间加到基本时间上。

Table A-4. 指令的分时, 37x 模板- 续

功能分类	功能模块	允许 37x	禁止 37x	增量 37x	容量
关系运算	等于 (INT)	1	0	–	10
	等于(DINT)	2	0	–	16
	等于(REAL)	5	0	–	14
	不等于(INT)	1	0	–	10
	不等于(DINT)	1	0	–	16
	不等于(REAL)	5	0	–	14
	大于 (INT)	1	0	–	10
	大于 (DINT)	1	0	–	16
	大于 (REAL)	5	0	–	14
	大于/等于(INT)	1	0	–	10
	大于/等于(DINT)	1	0	–	10
	大于/等于(REAL)	5	0	–	14
	小于 (INT)	1	0	–	10
	小于 (DINT)	1	0	–	16
	小于(REAL)	5	0	–	14
	小于/等于(INT)	1	0	–	10
	小于/等于(DINT)	3	0	–	16
	小于/等于(REAL)	5	0	–	14
	范围 (INT)	2	0	–	13
	范围 (DINT)	2	0	–	22
	范围 (WORD)	1	0	–	13
位操作 Operation	逻辑与 AND	2	0	–	13
	逻辑或 OR	2	0	–	13
	逻辑异或OR	1	0	–	13
	逻辑非, NOT	1	0	–	10
	左移位	5	0	1	16
	右移位	5	0	1	16
	循环左移	5	0	1	16
	循环右移	5	0	1	16
	定位	5	0	–	13
	位清除	5	0	–	13
	位检测	5	0	–	13
	置位	5	0	–	13
	屏蔽比较 (WORD)	9	0	–	25
	屏蔽比较 (DWORD)	10	0	–	25

注意:
1. 对于制表功能, 增量指定单元长度.; 低于位运算功能, 微秒.; 对于数据移动功能, 微秒/位数或字数.
2. 对于单独的单元长度类型 %R, %AI, 和 %AQ.
3. COMMREQ 使用 CPU 和 HSC测试.
4. DOIO为输出到离散输出模块时间 .
5. 有多种情况下,以上时间表示最坏情况下.
6. 对于含有增量的指令, (长度 –1)乘以增量时间加到基本时间上。

Table A-4. 指令的分时, 37x 模板- 续

功能分类	功能模块	允许 37x	禁止 37x	增量 37x	容量
数据传送	传送 (INT)	2	0	1	10
	传送 (BIT)	5	0	1	13
	传送(WORD)	2	0	1	10
	传送 (REAL)	5	0	1	13
	块传送(INT)	2	0	–	28
	块传送(WORD)	3	0	–	28
	块传送(REAL)	11	1	–	13
	块清除	1	0	1	11
	移位寄存器 (BIT)	10	0	1	16
	移位寄存器 (WORD)	15	0	1	16
	位定序器	14	10	1	16
	通讯请求	200	200	–	13
功能表	数组传送				
	INT	10	0	1	22
	DINT	15	0	1	22
	BIT	10	0	1	22
	BYTE	10	0	1	22
	WORD	10	0	1	22
	查寻相等				
	INT	5	0	1	19
	DINT	5	0	2	22
	BYTE	5	0	1	19
	WORD	5	0	1	19
	查寻不相等				
	INT	5	0	1	19
	DINT	10	0	2	22
	BYTE	5	0	2	19
	WORD	5	0	2	19
	查寻大于				
	INT	5	0	1	19
	DINT	5	0	2	22
	BYTE	10	0	1	19
	WORD	5	0	1	19
	查寻大于/等于				
	INT	5	0	1	19
	DINT	5	0	2	22
	BYTE	5	0	1	19
	WORD	5	0	1	19

注意:
1. 对于制表功能，增量指定单元长度.; 低于位运算功能，微秒.; 对于数据移动功能，微秒/位数或字数.
2. 对于单独的单元长度类型 %R, %AI, 和 %AQ.
3. COMMREQ 使用 CPU 和 HSC测试.
4. DOIO为输出到离散输出模块时间 .
5. 有多种情况下,以上时间表示最坏情况下.
6. 对于含有增量的指令, (长度 –1)乘以增量时间加到基本时间上。

Table A-4. 指令的分时, 37x 模板- 续

功能 分类	功能模块	允许 37x	禁止 37x	增量 37x	容量
	查寻小于				
	INT	5	0	1	19
	DINT	10	0	2	22
	BYTE	5	0	1	19
	WORD	5	0	1	19
	查寻小于/等于				
	INT	5	0	1	19
	DINT	5	0	2	22
	BYTE	5	0	1	19
	WORD	5	0	1	19
转换功能	转换到 INT	5	0	–	10
	转换到 BCD-4	5	0	–	10
	转换到REAL	5	0	–	8
	转换到WORD	5	0	–	11
	舍位到 INT	5	0	–	11
	舍位到 DINT	5	0	–	11
控制功能	调用子程序	15	0	–	7
	Do I/O	5	0	–	13
	PID – ISA 算法	14	10	–	16
	PID – IND 算法	14	10	–	16
	End 指令	–	–	–	–
	服务请求				
	#6	5	0	–	10
	#7 (Read)	10	0	–	10
	#7 (Set)	5	0	–	10
	#14	15	0	–	10
	#15	5	0	–	10
	#16	10	0	–	10
	#18	255	0	–	10
	#23	25	0	–	10
	#26//30**	155	0	–	10
	#29	5	0	–	10
	嵌套 MCR/ENDMCR 嵌套	1	0	–	4
	SER 8通道	60	0	=	
	SER 16通道	199	0	=	

**SVCREQ #26/30 使用高速计数器，16点输出，5槽机架测试。

注意:
1. 对于制表功能，增量指定单元长度.; 低于位运算功能，微秒.; 对于数据移动功能，微秒/位数或字数.
2. 对于单独的单元长度类型 %R, %AI, 和 %AQ.
3. COMMREQ 使用 CPU 和 HSC测试.
4. DOIO为输出到离散输出模块时间 .
5. 有多种情况下,以上时间表示最坏情况下.
6. 对于含有增量的指令, (长度 –1)乘以增量时间加到基本时间上。

CPU 布尔型执行时间

这个表列出了对于系列90-30CPU 模板执行线圈和触点的时间。

表 A-5. 布尔型执行时间

CPU 类型	执行时间 1/1,000 布尔型触点/线圈
模板 37x	0.15 毫秒
模板35x 和 36x	0.22毫秒
模板340/341	0.3毫秒
模板331	0.4毫秒
模板313/323	0.6毫秒
模板311	18.0毫秒

I CPU 350 – 374的指令大小

下表中存贮器的尺寸,是使用存贮器的字节数,它必须在梯形图编程应用程序中指令给定。

表 A-6. CPU 350 – 374指令大小

功能	存贮器尺寸
取出栈 AND	1
取出栈OR	1
复制栈	1
取出栈	1
初始化栈	1
标签	5
跳转	5
全部其他指令	3
功能块 – 见表 A-2	多样

附录B 说明故障表

系列90-30, 90-20, 和 90 Micro PLC都有两个故障表，I/O设备(I/O控制器)产生的I/O故障表和PLC内部故障的PLC故障表。本附录介绍的内容有助于用户在读取故障表时能够理解报文的结构形式，两个表中含有相似的信息，故障表中的数据仅存在PLC中，并不是在文件夹中，因此，如果使用Logicmaster，必须在ONLINE或MONITOR模式才能查看故障。

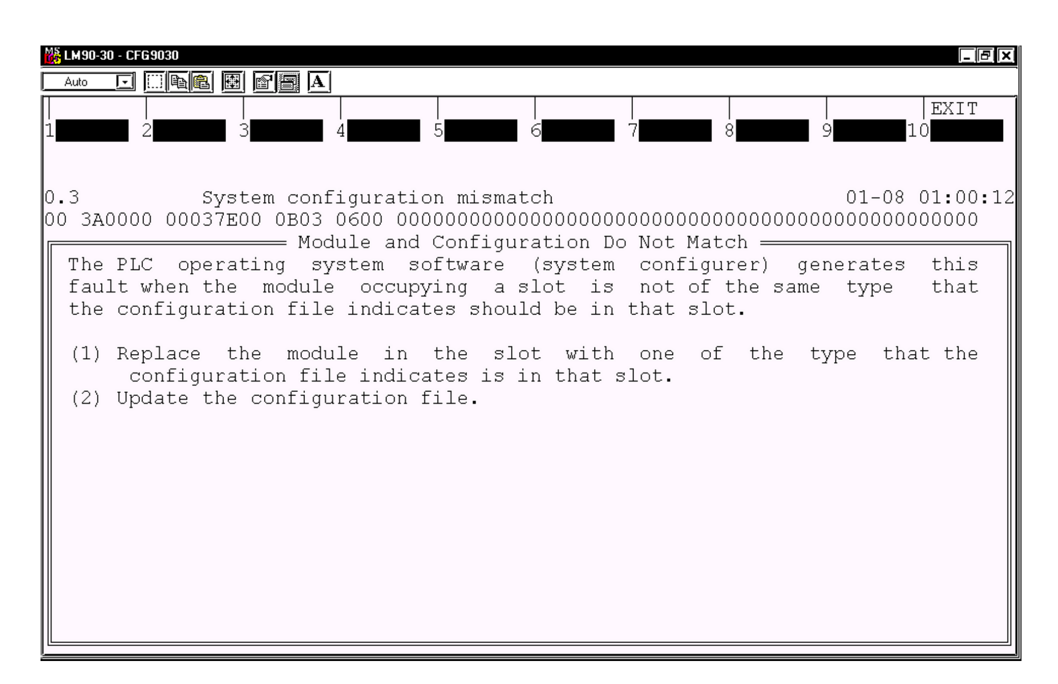
- PLC故障表内容:
   - Ü 故障位置.
   - Ü 故障描述.
   - Ü 故障日期和时间.
- I/O故障表内容:
   - Ü 故障位置.
   - † 基准地址.
   - † 故障等级.
   - † 故障类型.
   - † 故障日期和时间t.

PLC故障表

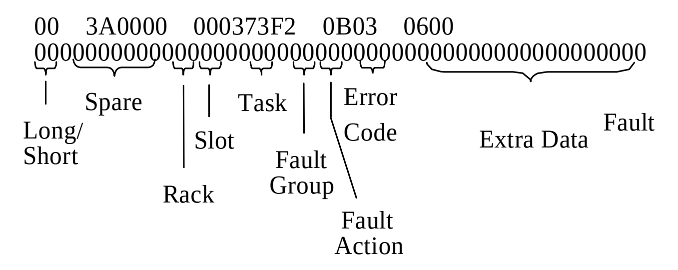
通过编程软件可以访问PLC故障表，对于关于访问故障表的信息，可以查阅在线帮助，*Logicmaster 90 系列90-30/20/Micro编程软件用户手册*，GFK-0466。

示例

下图显示了已经通过zoom键查看的System Configuration Mismatch（系统配置不当）故障的示例



下图表中是识别上System ConfigurationMismatch故障的每个故障区域。



System Configuration Mismatch 故障解释如下（所有数据以16进制表示）

区域	数值	注释
长/短	00	00=短，表示只有8个字节故障附加数据
机架	00	主机架（机架 0）
槽位	03	第3个槽位
任务	F2	
故障组	0B	系统配置不匹配故障
故障反应	03	致命故障
错码	06	
故障附加数据	00	在8个字节中无故障附加数据报表

下面几段对故障条目中的每个字加以说明，其中包括说明每个字段的数值范围的表格。

长/短指示项

该字节说明故障包含的故障附加数据是8个字节还是24个字节。

类型	数码	故障附加数据
短	00	8 字节
长	01	24 字节

备用

这6个字节为填充字节，用于把PLC故障表条目调整得和I/O故障表田亩一样长。

机架

机架号的范围是0到7，0表示主机架，包含PLC，机架1到7，是扩展机架通过扩展总线连接PLC。

槽位

槽位号从0到9，PLC CPU通常占用主机架（机架0）的第一槽。

任务

任务号范围从0到65535，有时任务号为PLC工程师提供的其他信息，通常，任务可以忽略。

PLC故障组

故障组是故障的最高分类，它不同于一般的故障分类。Logicmaster 90-30/20/Micro软件显示的故障说明文正是以故障组和错误代码为基础的。

表 B-1列出了PLC故障表中可能出现的故障组。

最后一个不能屏蔽的故障组，即***Additional PLC Fault Codes***（附加PLC故障码）。已经证明可以在PLC不必知道报警码的情况下处理系统中新的故障。所有不被识别的PLC型报警码都属于此故障组。

表 B-1. PLC故障组

号码			
十进制	**十六进制**	**组名**	**故障反应**
1	1	损坏，或丢失机架	致命性
4	4	损坏，或丢失人选模块	诊断性
5	5	增加，或附加机架	诊断性
8	8	增加，或附加人选模块	诊断性
11	B	系统配置失配	致命性
12	C	系统总线错误	诊断性
13	D	PLC CPU硬件故障	致命性
14	E	非致命模块硬件故障	诊断性
16	10	任选模块软件故障	诊断性
17	11	程序块检验和故障	致命性
18	12	低电池信号	诊断性
19	13	恒定扫描时间超出	诊断性
20	14	PLC系统故障表已满	诊断性
21	15	I/O 故障表已满	诊断性
22	16	用户应用程序故障	诊断性
–	–	附加PLC故障码	待确定
128	80	系统总线故障	致命性
129	81	用户程序未启动	提示性
130	82	检测到用户RAM故障	致命性
132	84	密码访问失败	提示性
135	87	PLC CPU软件故障	致命性
137	89	PLC 顺序储存故障	致命性

故障反应

每个故障的反应可以是与其有关的三种反应中的一种，这些故障反应在系列90-30 PLC中是固定的，用户不可以改变。

表 B-2. PLC故障反应

故障反应	CPU动作	代码
提示性	故障表中记录	1
2诊断性	故障表中记录故障 设置故障参考标号	2
致命性	故障表中记录故障 设置故障参考标号 进入停止模式	3

错码

错码进一步对故障作说明。每个故障组都有其自己的错误代码。表B-3显示了PLC软件错误码（组87H）的出错码。

表 B-3. PLC CPU软件故障报警错码

十进制	十六进制	名称
20	14	PLC程序内存故障
39	27	PLC程序内存故障
82	52	底板通讯故障
90	5A	用户关机请求
其他		PLC CPU内部系统错误

表 B-4 显示了所有其他故障的错码.

表 B-4.故障报警错码

十进制	十六进制	名称
任选模块组 (4)损失PLC错码		
44	2C	任选模块软复位故障
45	2D	任选模块软复位故障
255	FF	任选模块软通讯失败
79	4F	子板丢失
任选模块组***(8)***复位，增加和附加错码		
2	2	模板重新启动完成
04	4	增加子板
05	5	子板重新启动
	其他	任选模板的复位，增加或附加
任选模块(***10HEX)***软件故障错码		
1	1	不被支持的板类型
2	2	启动COMMREQ的输出信息电子邮箱满。e
3	3	COMREQ – 响应电子邮箱满e
5	5	底板与PLC通讯，丢失请求
11	B	资源 (分配溢出等.)错误
13	D	用户程序错误
401	191	模块软件故障，请求重装
系统配置失配组错码		
8	8	模拟扩展失配
10	A	不被支持特性
23	17	程序超出存储限度
58	3A	子板的失配
系统总线错误码		
其他		系统总线错误
程序块检验和组错码		
3	3	程序或程序块检验和失败
低电池信号错码		
0	0	PLC CPU 或其他模块电池故障
1	1	PLC CPU 或其他模块低电池故障
用户应用程序故障组错码		
2	2	PLC 监视计时器超时
5	5	COMREQ – WAIT方式对该命令无效
6	6	COMREQ – 坏任务ID
7	7	应用程序溢出
系统总线故障组错码		
1	1	操作系统
用户RAM上电故障错码		
1	1	用户RAM上电故障
2	2	检测到非布尔操作码
3	3	PLC_ISCP_PC_OVERFLOW
4	4	PRG_SYNTAX_ERR
PLC CPU硬件故障错码		
全码		PLC CPU 硬件错误

故障附加数据

该区域为故障条目的细节，下面的示例说明了哪些数据可以出现：

示例 – 被损坏的用户RAM组

系统配置失配组中的四个错误代码提供故障附加数据：

表 B-5. PLC故障数据 – 检测到的非布尔操作码

故障附加数据	型号不匹配
[0]	ISCP 故障寄存器内容
[1]	坏操作码
[2,3]	ISCP 程序计数器
[4,5]	功能码

对于PLC CPU中的RAM故障(PLC CPU硬件故障中的一个故障), 故障地址被存储在该字段的头4个字节中.

PLC 故障时间标记

PLC CPU 硬件故障(RAM 失败)

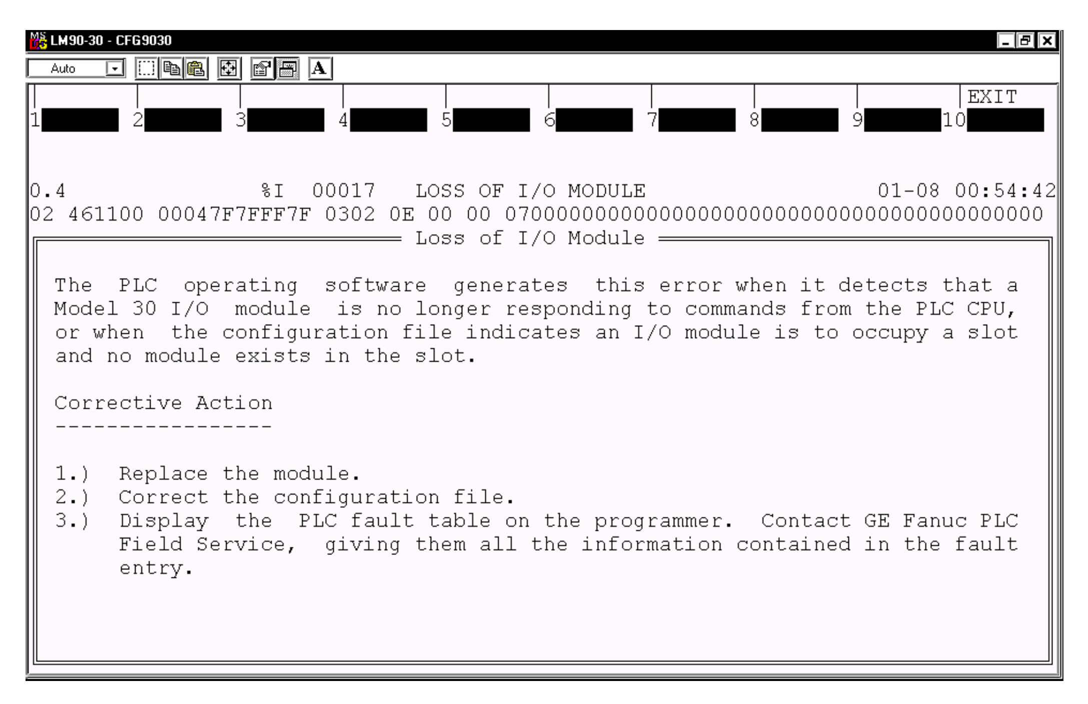
6个字节的时间标记为PLC CPU记录故障时的系统时钟的时间。（时间按BCD格式编码）

表 B-6. PLC 故障时间标记

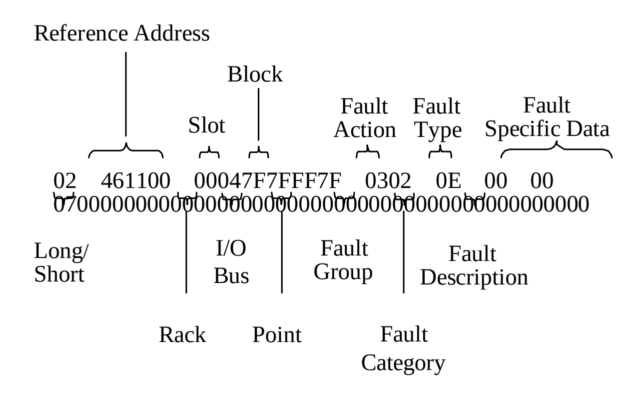
字节号	说明
1	秒钟
2	分钟
3	小时
4	日
5	月
6	年

I/O 故障表

下图显示的是通过ZOOM键显示的丢失I/O模块故障的示例。



下图表中是识别上System ConfigurationMismatch故障的每个故障区域。



下面的段落描述了I/O故障表中的每个区域，其中包括说明每个字段的数值范围的表格。

长/短指示项

该字节说明故障包含的故障附加数据是5个字节还是21个字节。

.
表 B-7. I/O 故障指示项类型

类型	代码	故障指定数据
短	02	5字节
长	03	21字节

参考地址

参考地址为3个字节，它包括I/O存储类型和内存中与故障点对应的位置（或偏置）。也就是说，当Genius块故障或积分模拟模块故障发生时，参考地址就是发生故障块中的第一个点。

表 B-8. I/O 参考地址s

字节	说明	范围
0	存储类型	0 – FF
1–2	偏置	0 – 7FF

存储类型字节为下面数值中的一种

表 B-9. I/O参考地址存储类型

名称	值 (十六进制)
模拟输入	0A
模拟输出	0C
模拟分组	0D
数字输入	10或 46
数字输出	12或 48
数字分组	1F

I/O 故障地址

I/O故障地址为6个字节，它包括机架，槽位，块和发生故障I/O点的点地址。点地址是一个字，所有其他地址都为一个字节。五个数值不能都出现在同一故障中。

当一个I/O故障地址中不包括5个地址，地址中则出现一个7F（十六进制）以表明有效位停止的地方。例如，如果总线字节中出现7F，那么故障位模块故障。但是，机架和槽位必须有效。

机架

机架号的范围是0到7，0表示主机架，包含PLC，机架1到7，是扩展机架通过扩展总线连接PLC。

槽位

槽位号从0到9，PLC CPU通常占用主机架（机架0）的第一槽。

故障点

故障点范围为1到1024（十进制），它说明当故障为点类型时模块上出现的故障的点。

.

I/O 故障组

故障组是故障的最高分类，它与一般的故障分类不同。Logicmaster 90-30/20/Micro软件显示的故障说明文本是以故障组和错码为基础的。

表 B-10列出了I/O故障表中可能出现的故障组。小于80（十六进制）的组号为屏蔽比较故障。

最后一个不可屏蔽的故障组“附加I/O故障码”已证明在PLC不必知道报警码情况下可以解决系统中的新故障。所有不被识别的I/O型报警码都属于这一组。

表 B-10. I/O故障组

组号	组名	故障反应
3	损失，或丢失，I/O模块	诊断性
7	增加，或附加，I/O模块	诊断性
9	IOC 或 I/O总线故障	诊断性
A	I/O 模板故障	诊断性
–	附加I/O故障码	待确定

I/O 故障反应

故障反应指的是当故障发生时，PLC CPU应有的什么样的反应，表B-11列出了可能出现的故障反应。

表 B-11. I/O 故障反应

故障反应	CPU 反应	代码
提示性	故障表中记录故障	1
诊断性	故障表中记录故障 设置故障参考标号	2
致命性	故障表中记录故障 设置故障参考标号 进入STOP模式	3

I/O 故障具体数据

一个I/O故障条目最多可以包括5个字节的I/O故障具体数据。

符号故障具体数据

表 B-12列出了块电路配置所需的数据.

表 B-12. I/O故障具体数据

十进制数	十六进制码	说明
电路配置		
	1 2 3	电路为三态输入 电路为输入 电路为输出

具体故障的故障反应

强制/非强制电路故障按提示故障报告，其他故障为诊断故障或致命故障。

型号不匹配，I/O类型不匹配和不存在I/O模块等故障报告在“系统配置不匹配”组下的PLC故障表中，而不报告在I/O故障表中。

I/O故障时间标记

6个字节的时间标记是PLC CPU记录故障时系统时钟的时间。时间按BCD格式编码。

表 B-13. I/O 故障时间标记

字节号	说明
1	秒钟
2	分钟
3	小时
4	日期
5	月份
6	年份

附录 C 指令助记符

在程序显示/编辑模式中，可以通过&符号和指令助记符，很快引入或查寻一个编程指令。对于一些指令，也可指定一参考地址或别名、一个标号或一个存贮单元参考地址。

这一附录中列出了系列Logicmaster的 90-30/20/Micro编程软件中的编程指令助记符。全部助记符列于表的第三栏，每一指令的最短项列于第四栏。

在编程的任何时候，都可以按下ALT和L键，借助屏幕已显示这些助记符。

功能分类	指令	助记符						
		All	INT	DINT	BIT	BYTE	WORD	REAL
触点	任何触点	&CON	&CON					
	常开触点	&NOCON	&NOCON					
	常闭触点	&NCCON	&NCCON					
	延续触点	&CONC	&CONC					
线圈	任何线圈	&COI	&COI					
	常开线圈	&NOCOI	&NOCOI					
	求反线圈	&NCCOI	&NCCOI					
	正向跳变线圈	&PCOI	&PCOI					
	反向跳变线圈	&NCOI	&NCOI					
	置位线圈	&SL	&SL					
	复位线圈	&RL	&RL					
	保持置位线圈	&SM	&SM					
	保持复位线圈	&RM	&RM					
	保持线圈	&NOM	&NOM					
	求反保持线圈	&NCM	&NCM					
	延续线圈	&COILC	&COILC					
链路	水平链路	&HO	&HO					
	垂直链路	&VE	&VE					
定时器	接通延时定时器	&ON	&ON					
	逝去延时定时器	&TM	&TM					
	断开延时定时器	&OF	&OF					
计数器	加计数器	&UP	&UP					
	减计数器	&DN	&DN					

功能分类	指令	助记符							
		All	BCD-4	INT	DINT	BIT	BYTE	WORD	REAL
数学运算	加	&AD		&AD_I	&AD_DI				&AD_R
	减	&SUB		&SUB_I	&SUB_DI				&SUB_R
	乘	&MUL		&MUL_I	&MUL_DI				&MUL_R
	除	&DIV		&DIV_I	&DIV_DI				&DIV_R
	模 除	&MOD		&MOD_I	&MOD_DI				&MOD_R&SQ_R
	平方根	&SQ		&SQ_I	&SQ_DI				
	正弦	&SIN							
	余弦	&COS							
	正切	&TAN							
	反正弦	&ASIN							
	反余弦	&ACOS							
	反正切	&ATAN							
	LG	&LOG							
	自然对数e	&LN							
	E次方	&EXP							
	X次方	&EXPT							
关系运算	等于	&EQ		&EQ_I	&EQ_DI				&EQ_R
	不等于	&NE		&NE_I	&NE_DI				&NE_R
	大于	&GT		&GT_I	&GT_DI				&GT_R
	大于或等于	&GE		&GE_I	&GE_DI				&GE_R
	小于	&LT		&LT_I	&LT_DI				&LT_R
	小于或等于	&LE		&LE_I	&LE_DI				&LE_R
位操作	与AND	&AN						&AN_W	
	或OR	&OR						&OR_W	
	异或OR	&XO						&XO_W	
	非 NOT	&NOT						&NOT_W	
	左移位	&SHL						&SHL_W	
	右移位	&SHR						&SHR_W	
	循环左移	&ROL						&ROL_W	
	循环右移	&ROR						&ROR_W	
	位测试	&BT						&BT_W	
	位置位	&BS						&BS_W	
	位清除	&BCL						&BCL_W	
	定位	&BP						&BP_W	
	屏蔽比较	&MCMP						&MCM_W	
转换	转换乘整数	&TO_INT	&TO_INT_BCD4						
	转换成双整数	&TO_DINT							&BCD4_R
	转换成 BCD–4	&BCD4			&TO_REAL_DI				
	转换成实数	&TO_REAL						&TO_REAL_W	
	转换成字	&TO_W							
	舍位为整数	&TRINT							
	舍位为双整数	&TRDINT							

功能分类	指令	助记符						
		All	**INT**	**DINT**	**BIT**	**BYTE**	**WORD**	**REAL**
数据传送	传送	&MOV	&MOV_I		&MOV_BI		&MOV_W	&MOV_R
	块传送	&BLKM	&BLKM_I				&BLKM_W	&BLKM_R
	块清除	&BLKC						
	移位寄存器	&SHF			&SHF_BI		&AR_W	
	位定序器	&BI						
	通讯请求	&COMMR						
表功能	数组传送	&AR	&AR_I	&AR_DI	&AR_BI	&AR_BY	&AR_W	
	查寻相等	&SRCHE	&SRCHE_I	&SRCHE_DI		&SRCHE_BY	&SRCHE_W	
	查寻不相等	&SRCHN	&SRCHN_I	&SRCHN_DI		&SRCHN_BY	&SRCHN_W	
	查寻大于	&SRCHGT	&SRCHGT_I	&SRCHGT_DI		&SRCHGT_BY	&SRCHGT_W	
	查寻大于或等于	&SRCHGE	&SRCHGE_I	&SRCHGE_DI		&SRCHGE_BY	&SRCHGE_W	
	查寻小于	&SRCHLT	&SRCHLT_I	&SRCHLT_DI		&SRCHLT_BY	&SRCHLT_W	
	查寻小于或等于	&SRCHLE	&SRCHLE_I	&SRCHLE_DI		&SRCHLE_BY	&SRCHLE_W_	
控制功能	调用子程序	&CA						
	Do I/O	&DO						
	SER	&SER						
	PID – ISA 算法	&PIDIS						
	PID – IND 算法	&PIDIN						
	SFC复位	&SFCR						
	结束	&END						
	回路解释	&COMME						
	系统服务请求	&SV						
	主令控制继电器	&MCR						
	结束主令控制继电器	&ENDMCR						
	嵌套主令控制继电器	&MCRN						
	嵌套结束主令控制器	&ENDMCRN						
	跳转	&JUMP						
	嵌套跳转	&JUMPN						
	标号	&LABEL						
	嵌套标号	&LABELN						

附录x D

键功能

该附录列出了软件环境下起作用的键盘功能，这一信息也可通过按ALT-K键在编程器屏幕上显示出来。

键序列	说明	键序列	说明
全部软断开延时定时器件包有效的功能键			
ALT-A	中止	CTRL-Break	退出软件包
ALT-C	清字段	Esc	退出窗口.
ALT-M	改变编程器方式 .	CTRL-Home	前一指令行.
ALT-R	改变PLC运行/停止状态 .	CTRL-End	下一指令行.
ALT-E	触发状态区	CTRL- <—	在工作区内左移光标
ALT-J	触发指令行.	CTRL-—>	在工作区内右移光标.
ALT-L	列文件目录 .	CTRL-D	减量参考地址
ALT-P	打印屏幕.	CTRL-U	增量参考地址
ALT-H	帮助	Tab	改变/增加文件内容
ALT-K	键帮助.	Shift-Tab	改变/减少文件内容.
ALT-I	指令助记符帮助 .	Enter	接受文件内容
ALT-N	触发显示选择项.	CTRL-E	显示最后的系统误差.
ALT-T	其实引导模式.	F12 or Keypad -	转换离散参考地址
ALT-Q	停止引导模式.	F11 or Keypad *	超驰离散参考地址
ALT-n	返回文件 n (n = 0 到 9).		
只在程序编辑中有效的键			
ALT-B	触发文件编辑器铃.	Keypad +	接受回路.
ALT-D	删除回路单元/删除回路 .	Enter	接受回路.
ALT-S	向PLC中存贮模块 .	CTRL-PgUp	前一回路
ALT-X	显示窗口大小	CTRL-PgDn	下一回路
ALT-U	刷新磁盘	~	水平分路
ALT-V	变量表窗口	|	垂直分路
ALT-F2	进入操作参考表	Tab	进入下一操作区.
特殊键			
ALT-O	口令超控，只用于设置软件的口令屏幕。		

下页帮助卡片包含Logicmaster 90-30/20/Micro 软件键帮助和指令助记符列表。该卡片分成三列而且打孔容易从本手册中取下来。

GFK-0467M

Print side 1 of GFJ-055D on this page.

Appendix D Key Functions

附录 E 使用浮点数

用户在使用浮点数时必须了解几个要点，本附录的第一部分将对这几个要点作一介绍。

引入和显示浮点数的说明，参看页E-5及后面部分。

注意

仅在35x和36x 系列CPU，9.00版本或更高版本和CPU352、37X系列所有版本支持浮点数。.

浮点数

编程软件提供编辑、显示、存贮和实数值检索能力，一些功能在浮点数上运行，然而编程软件使用浮点数，必须提供一个35x, 36x 或 37x 系列 CPU (见上注意)，浮点数用十进制表示，并显示6个有效位。

注意

在本手册中，“浮点”和“实数”互换使用来说明编程软件的浮点数显示和引入特性。

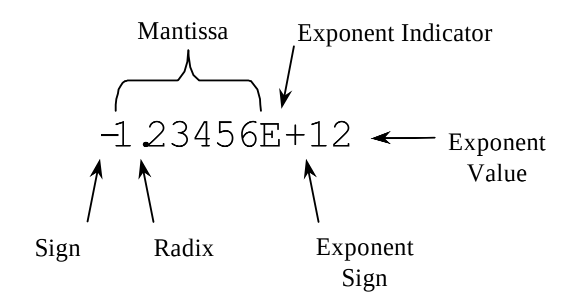
采用以下的格式，对于9999999到0.0001之间的数，不显示指数和大于6或7的有效数字。例如：

引入	显示	说明
.000123456789	+.0001234567	10个数字，6或7个有效数字
–12.345e-2	–.1234500	7个数字，6或7个有效数字
1234	+1234.000	7个数字，6或7个有效数字

超出上述范围，只显示6个有效数字并且显示格式为: +1.23456E+12

实数术语

一个实数存贮在32位的双字寄存器中。以下讨论用于实数各部分的术语。



符号 – 正号或者负号，存贮在双字的最高有效位（32位），在位32中1表示负数，0表示正数。

点 –点符号，把整个数值分成两部分，指数和尾数。对于十进制数，通常叫做小数点。

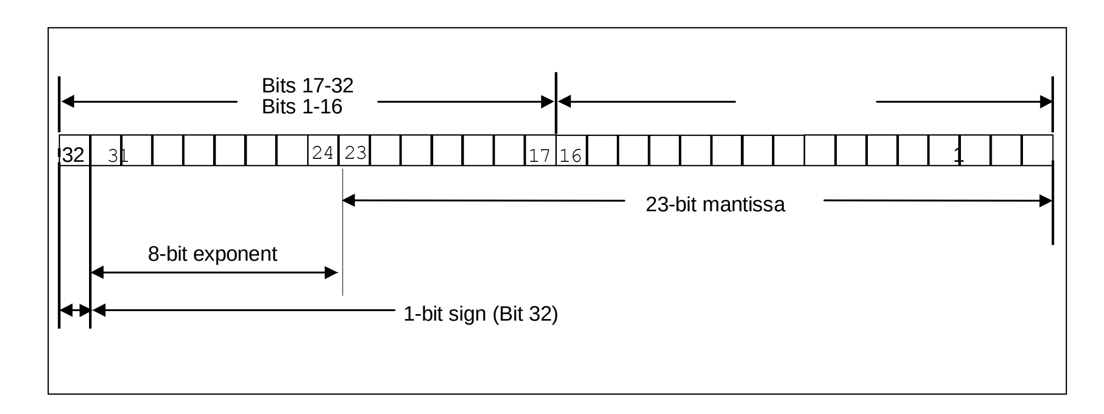
指数 – 8位, 32-位双字的第31位到第24位. 数值范围 +127 到 –126; 然而，它总是正数因为CPU自动对该值加127。

尾数 –基数不含符号位和指数位。共23位，32-位双字的第1位到第23位。

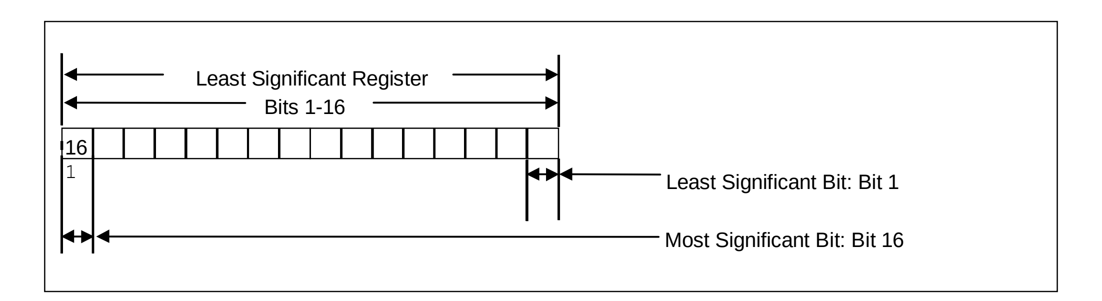
精度 – 和数值有效位相关. 双精度带符号寄存器使用31个位存储数值（32位用作符号位），它的精度高于实数（浮点数），浮点数只使用23位来存储尾数。

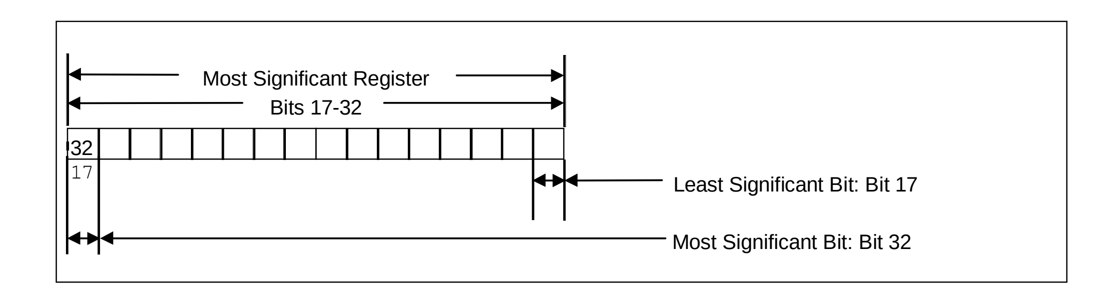
浮点数的内部格式

浮点数以单精度IEEE标准格式存储。这种格式要求32位，并且传送到两个相邻的16位PLC寄存器中，位的编码用下图表表示：



单浮点数使用的寄存器用下图表示如下，在下图中，如果浮点数占用寄存器%R0005和%R0006，例如，%R0005 是最低有效寄存器，%R0006是最高有效位寄存器。





浮点数值

下表用于计算存储在两个寄存器中的二进制浮点数的值。

指数 (e)	尾数 (f)	浮点数的值
255	非零	不是一个有效数值(NaN).
255	0	$-1^{s} * \infty$
0 < e < 255	任何数	$-1^{s} * 2^{e-127} * 1.f$
0	非零	$-1^{s} * 2^{-126} * 0.f$
0	0	0

f = 尾数. 是一个二进制的小数
e = 指数. 该指数是一个整数E, E+127 is the power of 2 by which the mantissa must be multiplied to yield the floating-point value.
s = 符号位.
* = 乘法操作符.

例如，假定有浮点数12.5，其IEEE 浮点二进制表示法为：

```
01000001 01001000 00000000 00000000
```

或41480000十六进制,最高有效位(符号位)是0(s＝0),.后面8位最高有效位是10000010,
或130 十进制 (e=130).

尾数是按十二进制数存贮的，其十进制点接23位中的最高有效位。因此，尾数中的最高有效位是2-1的倍数。下一个最高有效位是2-2的倍数，以此类推，最低有效位为2-23的倍数，最后的23位（尾数）为：

```
1001000 00000000 00000000
```

则尾数的值为.5625 (即, $2^{-1} + 2^{-4}$).

因为 e > 0 而 e < 255, 所有采用前面表的第3个式子：:

$$
\begin{aligned}
\text{浮点数} &= -1^{s} * 2^{e-127} * 1.f \\
&= -1^{0} * 2^{130-127} * 1.5625 \\
&= 1 * 2^{3} * 1.5625 \\
&= 8 * 1.5625 \\
&= 12.5
\end{aligned}
$$

因此，可以看到上述的二进制表示法是正确的。

可以用这种格式存储的浮点数为± 1.401298E–45 到± 3.402823E+38 之间的数和0.

引入和显示浮点数

在尾数中，最多可以引入和存储6或7个最高有效位，然而，编程软件将只显示其中的头6个位，尾数后可接一个正或负符号，如果没有符号，浮点数就被认为是正。

如果引入指数，其后应接字母E或e，而且尾数必须包括一个十进制点，以免将其误认为十六进制数，指数后可跟一个符号，如果无符号，则被认为是正。如果无指数引入，则被认为是0。浮点数中不允许有空位。

为了方便使用，命令行和字段数据条目中可接受几种格式。这些格式包括正数、十进制和跟指数的十进制数。一旦用户已引入数据和按Enter键，这些数就被转换为一个标准格式。

有效浮点数条目及其标准化显示的例子说明如下：

.

引入	编程器显示
250	+250.0000
+4	+4.000000
–2383019	–2383019.
34.	+34.00000
–.0036209	–.003620900
12.E+9	+1.20000E+10
–.0004E–11	–4.00000E–15
731.0388	+731.0388
99.20003e–29	+9.92000E–28

有效浮点数条目的举例说明如下：

有效条目	解释/结果
–433E23	遗漏十进制点　LM90显示信息“Bad numeric value.”
10e-19	遗漏十进制点　LM90显示信息“Bad numeric value.”
1 0.e19	在尾数中1和0之间有空格，实数引入必须尾数在位与字符之中不能有空格。Logicmaster 认为这个引入为错误的值+1.000000。
4.1e 19	指数中e和19之间存在空格，实数在位和字符之间不能有空格，Logicmaster 认为这个引入为错误的值+4.100000。

浮点数误差和操作

正和负无穷大

对于352或 374 CPU，当REAL功能产生数值大于3.402823E+38 或小于-3.402823E+38时，就会出现溢出。对于其他支持浮点数操作的90-30模板的范围是大于2^{16} 或小于 -2^{16}.，当数值超出这个范围时，功能的OK输出置为OFF，并且结果设为正无穷大 (对于352CPU或374CPU大于3.402823E+38或所有类型中大于2^{16} 数值)或负无穷大(所有类型中小于–3.402823E+38 或 -2^{16}数值)。用户可通过检测有无正常输出来确定出现溢出的地方。

尾数	梯形图值	参考表值(十六进制)	说明
POS_INF	+OVERFLOW	7F80 0000	IEEE 十六进制中表示正无穷大
NEG_INF	-OVERFLOW	FF80 0000	IEEE 十六进制中表示负无穷大.

注意

如果使用软件浮点数(所有支持浮点操作的类型除了352 或374 CPU),
在±1.175494E–38的数舍位到零 (0)
.

如果溢出产生的无穷大作为其他REAL功能的操作数，则导致不确定的结果，这不确定的结果定义为NAN（非数值），例如，正无穷大加负无穷大的结果就是不确定的。当用正无穷大和负无穷大启用ADD REAL功能时，产生的结果就是一个NAN.

非数值 (NaN)

非数值是一个不确定的数，像0被0除的结果， 正负无穷大不被认为是NAN,下面部分将帮助识别什么时候产生NAN结果。

352或374型CPU的NaN代码

在352或374型CPU里,能够产生一个NAN的每一个REAL功能可产生一个能够识别该功能的专用化NAN并能够在应用参考表中读取。在 Logicmaster梯形逻辑窗口将显示无符号术语“OVERFLOW.” (如果在 “OVERFLOW”前加上正或负号,它显示正或负无穷大.)

352和374 型CPU非数值 (NaN) 代码		
尾数	参考表 值（十六进制）	说明
NaN_ADD.	7F81 FFFF	实数加误差值，十六进制
NaN_SUB	7F81 FFFF	实数减误差值，十六进制.
NaN_MUL	7F82 FFFF	实数乘误差值，十六进制.
NaN_DIV	7F83 FFFF	实数除误差值，十六进制
NaN_SQRT	7F84 FFFF	实数平方根误差值，十六进制.
NaN_LOG	7F85 FFFF	实数对数误差值，十六进制
NaN_POW0	7F86 FFFF	实数指数误差值，十六进制
NaN_SIN	7F87 FFFF	实数正弦误差值，十六进制
NaN_COS	7F88 FFFF	实数余弦误差值，十六进制
NaN_TAN	7F89 FFFF	实数正切误差值，十六进制
NaN_ASIN	7F8A FFFF	实数反正弦误差值，十六进制.
NaN_ACOS	7F8B FFFF	实数反余弦误差值，十六进制.
NaN_BCD	7F8C FFFF	BCD-4到实数误差
REAL_INDEF	FFC0 0000	不定实数，0除0误差.

35x, 36x, 和37x 型CPU的NAN代码(除352 CPU)

固件支持浮点数操作的所有系列90-30 CPU (除了352型 CPU, 硬件基础) 仅仅产生一个NaN输出： FFFF FFFF. 表明在Logicmaster梯形逻辑窗口将显示无符号“OVERFLOW.”

35x, 36x,和37xCPU (除352 CPU)的非数值 (NaN) 类型		
尾数	参考表 值（十六进制）	说明
NaN_SW	FFFF FFFF	所有NaN软件浮点代码

NAN和无穷大数的传播和通流

当NAN结果反馈到另一个功能时，它将通过到结果，例如，一个NAN_ADD对于SUB REAL功能是第一个操作数，SUB_REAL功能结果是NAN_ADD,如果对于一个功能的操作数都是NAN，第一个操作数将通过，因为这个结果通过功能传播，这将能够识别在什么地方产生NAN。

注意

对于NaN，输出OK为OFF .

下表说明对无穷大数进行加法，乘法等运算时是否流过电流。如上所述，输出会超过正负范围即正无穷大或负无穷大。

表 E-1. 浮点数数学运算

操作	输入 1	输入 2	输出	得电
所有	数字	数字	正或负无穷大	No
除了DIV	无穷大	数字	无穷大	Yes
所有	数字	无穷大	无穷大	Yes
DIV	无穷大	数字	无穷大	No
所有	数字	数字	NaN	No

附录 F

编程软件比较

本手册为了Logicmaster (DOS基础的编程软件)用户编写，基于Windows基础的PLC软件产品，如CIMPLICITY® ME逻辑开发和 VersaPro®,软件内置在线帮助系统相对于手册提供更多指令集信息。 基于Windows编程软件用户应该知道指令在两种编程软件下看起来好像不同，但是实际上在PLC中它们作用一样。基于Windows编程软件的在线帮助系统能提供更准确指令信息。

除了在线帮助,你可以参考以下软件使用手册:

VersaPro™ 编程软件用户手册, GFK-1670

CIMPLICITY® Machine Edition手册, GFK-1868

注意

DRUM 指令支持

CPUs 350-364 版本 10.00 及以后，和CPU37x所有版本都支持该指令，Logicmaster任何版本都不支持DRUM指令；因此本手册不讨论。VersaPro, 版本1.1及以后，和Logic Developr所有版本支持该指令。 以上两个软件包中在线帮助中可找到该指令相关信息。

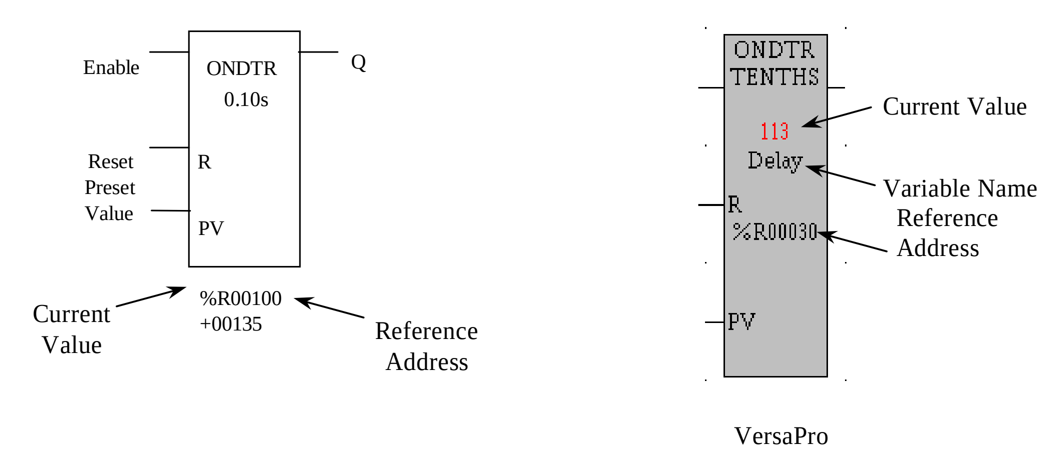
程序开始和结束标记

程序开始和结束标记用于Logicmaster编程软件中，但是基于Windows编程软件中不可见。

指令控制字地址区域

某些指令,例如定时器，计数器,和位定序器功能需要一组连续的字存储内部控制字。这组字通常叫做控制块。
在Logicmaster软件梯形图逻辑窗口中，控制块第一个字在指令下方(如下图%R00100位置). 对于基于Windows用户，控制块第一个字位于指令内部(如下图 %R00030 位置)。VersaPro 还在指令内部显示变量名称。

(如Delay) 。 如果未指定变量名，地址本身就是缺省的变量名(因此变量地址出现在指令内部两个位置上). VersaPro 中Delay右上角是数值113，代表当前变量值。



VersaPro

Logicmaster

实数显示区别

:方式不同，例如除以0运算。本手册附录E 讨论Logicmaster 在梯形图窗口和变量表中如何显示这些结果。

3

A

B

C

D

E

F

M

N

O

P

R

T

U





